Program Reversal Schedules for Single- and Multi-processor Machines

Dissertation

von

Andrea Walther

Dresden 1999

Program Reversal Schedules for Single- and Multi-processor Machines

DISSERTATION

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

vorgelegt

der Fakultät Mathematik und Naturwissenschaften
der Technischen Universität Dresden

von

Dipl.-Wirtschaftsmath. Andrea Walther

geboren am 9. September 1970 in Bremerhaven

Gutachter:	Prof. Ph.D. Andreas Griewank
	Prof. Ph.D. Richard P. Brent
	Ph.D. Alan Carle
Eingereicht am:	15. Oktober 1999
Tag der Verteidigung:	10. Dezember 1999

My attention was drawn to this research topic by Prof. Andreas Griewank, whose inspiration and advise formed the foundation of this thesis. Above all I would like to thank him for his support and many fruitful discussions.

I am indebted to Angela Giampietro, Uwe Naumann, and Olaf Vogel for helping me to put this thesis into shape. Special thanks are due to my husband for his aid during difficult times. Furthermore I am grateful to my parents for making my studies possible and for their encouragement.

Moreover I wish to thank my colleagues at the Institute of Scientific Computing, Technical University Dresden, for the pleasant working atmosphere. Last but not least thanks to the DFG for financial support within the research group "Identifikation und Optimierung komplexer Modelle auf der Basis analytischer Sensitivitätsberechnungen".

Contents

List of Figures

Notations

$a \mathrel{+}= b$	$a = a + b$
$\lceil w \rceil$	Smallest integer greater than or equal to $w \in \mathbb{R}$
$\lfloor w \rfloor$	Greatest integer less than or equal to $w \in \mathbb{R}$
b	Number of linear arguments (Def. 2.1, Page 11)
$\beta(c, r)$	Binomial coefficient $\binom{c+r}{c}$ (Def. 3.2, Page 27)
c	Total number of checkpoints available
c_k^j	Number of checkpoints used in the jth computational cycle for fixed $\hat{t}$ and a given parameter k
$c_{k,\hat{t}}^j$	Number of checkpoints used in the jth computational cycle for given parameters $\hat{t}$ and k
$\eta(c, r)$	$\equiv (\beta(c, r) + \beta(c-1, r))q + (\beta(c-1, r) - 1)b$ (Theo. 3.2, Page 34)
F	Evaluation procedure, evolutionary system
F_i	Physical step (Section 2.1, Section 2.2)
$\hat{F}_i$	Recording step (Section 2.1, Section 2.2)
$\bar{F}_i$	Reverse step (Section 2.1, Section 2.2)
$\gamma(c, r)$	$\equiv \beta(c, r)q + (\beta(c-1, r) - 1)b$ (Lemma 3.3, Page 27)
k	Number of available resources, each of which can be used either as processor or as checkpoint
l	Number of physical steps to be reversed
$\hat{l}$, $\tilde{l}$	Place of the next checkpoint
l_k	Maximal number of physical steps to be reversed for fixed $\hat{t}$ and a given parameter k
$l_{k,\hat{t}}$	Maximal number of physical steps to be reversed for given parameters $\hat{t}$ and k
$\mathbb{N}$	Set of natural numbers including zero
p	Number of processors available
p_k^j	Number of processors used in the jth computational cycle for fixed $\hat{t}$ and a given parameter k
$p_{k,\hat{t}}^j$	Number of processors used in the jth computational cycle for given parameters $\hat{t}$ and k
$P(i, h, c)$	Minimal partition function (Def. 3.3, Page 49)

$\phi(c,r)$	$\equiv \beta(c,r-1)+\beta(c-1,r-1)$ (Theo. 3.2, Page 34)
q	Number of previous states the new one depends on
r_i	Number of times F_i is performed while executing the reversal schedule S (Def. 2.4, Page 16)
$R(S)$	Resource profile of S (Def. 4.3, Page 61)
$\mathbb{R}$	Set of real numbers
s_k^j	Number of resources used in the jth computational cycle for fixed $\hat{t}$ and a given parameter k
$s_{k,\hat{t}}^j$	Number of resources in the jth computational cycle for given parameters $\hat{t}$ and k
$s(l,c)$	Minimal number of checkpoint writings to reverse l physical steps with c checkpoints on a serial machine (Theo. 3.2, Page 34)
S	Reversal schedule
$\tilde{S}$, $\bar{S}$	Auxiliary reversal schedules
S_k	Parallel reversal schedule using k processors and checkpoints for fixed $\hat{t}$
$S_{k,\hat{t}}$	Parallel reversal schedule using k processors and checkpoints for $\hat{t}$
t_i	Time needed to perform F_i (Section 2.3, Page 13)
$\bar{t}_i$	Time needed to perform $\bar{F}_i$ (Section 2.3, Page 13)
$\hat{t}_i$	Time needed to perform $\hat{F}_i$ (Section 2.3, Page 13)
$\bar{t}$	Time needed to perform a reverse step
$\hat{t}$	Time needed to perform a recording step
$t(i,h,c)$	Minimal cost to reverse the physical steps $F_i,\ldots,F_{h-1}$ with c checkpoints on a serial machine (Def. 3.1, Page 22)
$t(h,c)$	Minimal cost to reverse h physical steps with uniform step costs using c checkpoints on a serial machine (Theo. 3.1, Page 29)
T	Reversal trace (Def. 2.6, Page 17)
$\mathbb{Z}$	Set of integers

Chapter 1

Introduction

"Detailed studies of the real world impel us, albeit reluctantly, to take into account of the fact that the rate of change of physical systems depends not only on their present state, but also on their past history"
R. Bellmann & K.L. Cooke [BC63]

The mathematical specification of many applications involve nonlinear vector functions

$$F : \mathbb{R}^n \to \mathbb{R}^m, \qquad x \mapsto F(x),$$

that are evaluated by a computer program. For several purposes one may need to reverse the execution of the evaluation procedure F. The reversal of F is extensively used for example to calculate adjoints, to adapt parameters in a given model, or to steer a production process. Hence there are several contributions on weather data assimilation (e.g. [Tal91], [GC96]) or the optimization of production processes (e.g. [KW98]) dealing with this technique. Here the desired gradients can be obtained with a low temporal complexity by integrating the linear co-state equation backward. Therefore the reversal of F may be needed. This well-known technique is closely related to the reverse mode of algorithmic differentiation (AD) [Evt91], also called automatic or computational differentiation. For this mode of AD it is necessary to reverse the evaluation procedure F (e.g. [BBCG96], [Gri00]). Moreover, debugging and interactive control may require the reconstruction of previous states by some form of running the program that evaluates F backward. Serious difficulties arise when the simulated process described by F is not invertible or ill conditioned. In these cases one cannot simply apply the inverse function F^{-1}. Therefore the reversal of a program execution has received some, but only perfunctory, attention in the computer science literature (see e.g. [vdS93]). Nevertheless Bennett conjectured already in 1973 that a logarithmic growth in the spatial complexity might be achievable [Ben73].

An obvious way to reverse the evaluation of F is given by first recording the complete *execution log* onto a corresponding data structure called *tape* and subsequently reading the tape backward. The execution log contains for each arithmetic operation the operator and the addresses of the arguments. This basic approach causes a memory requirement proportional to the run time needed to evaluate F. Therefore the practical applicability is limited despite the availability of constantly growing memory systems.

Another way to reverse the calculation of F employs only a fixed and usually small amount of memory to store intermediate states called checkpoints or snapshots during the evaluation process. Then the execution log is generated piecewise by restarting the evaluation repeatedly from the suitably placed checkpoints. In this way the same values are computed several times according to requests by the reversal process. Applying this checkpoint technique the calculation of F can be reversed even in such cases, where the basic approach fails due to excessive memory requirement (e.g. [VO85], [Kub98] with respect to AD).

Throughout this thesis it is assumed that the evaluation of F comprises the evaluation of subfunctions F_i, $i = 0, \ldots, l-1$, called physical steps that act on state i to calculate the intermediate state $i+1$ for $i = 0, \ldots, l-1$. Hence F describes an *evolutionary system* as depicted in Fig. 1.1. Here, the

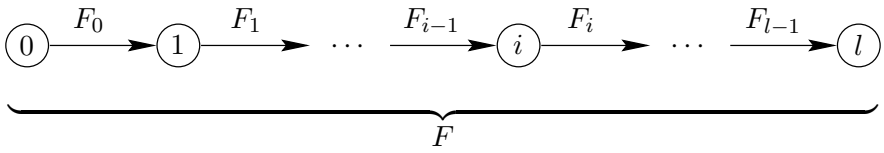


Figure 1.1: Evaluation of F

states represented by the counter i should be thought of as vectors of large dimensions representing the intermediate states of the evolutionary system F. The physical steps F_i describe mathematical mappings that in general cannot be reversed at a reasonable cost. Hence it is impossible to apply simply the inverses F_i^{-1} in order to run backward from state l to state 0. Also it will be assumed that due to their size only a certain number of intermediate states can be kept in memory.

If F describes an explicitly time-dependent problem or a general stepwise evaluation process each propagation of the state vector from state i to state $i+1$ can be seen as physical step F_i. In the case of an arbitrary function evaluation procedure written in an imperative programming language the execution can be interpreted as a time-dependent problem by combining an appropriate number of successively executed statements into one physical step. Therefore it causes no loss of generality basically to assume that the evaluation of the given computer program equals the computation of F_i, $i = 0, \ldots, l-1$.

Furthermore it is assumed that for each $i \in \{0, \ldots, l-1\}$ there exist functions $\hat{F}_i$ that cause the recording of the data required during the evaluation of F_i onto the tape and corresponding functions $\bar{F}_i$ that perform the reversal of the ith physical step using the tape. Having F_i, $\hat{F}_i$, and $\bar{F}_i$ at hand the

checkpoints can be thought of as pointers to nodes representing intermediate states i. Then all reversal strategies that use checkpoints perform the following *actions* (e.g. [Gri92], [GPRS96]) in order to reverse the execution of F:

a: Initialization: Reserve space for c checkpoints and copy the initial state 0 to the first one.

b: Forward sweep: Starting from a state i advance to a state j by performing the physical steps F_h, $h = i, \ldots, j-1$, without recording the execution log.

c: Recording step: Starting from state i perform recording step(s) to the currently final state by writing the execution log onto the tape.

d: Reverse Step: Perform a corresponding number of reverse steps from the currently final state to state i using the tape. Now state i becomes the final state. If the new final state is stored in a checkpoint, free the corresponding checkpoint up.

e: Checkpoint writing: Copy a state into a checkpoint.

f: Checkpoint reading: Read a state from a checkpoint.

It will be supposed that the time needed for writing or reading a checkpoint is negligible in comparison to the execution of one physical step.

On a serial machine only one processor is available to perform Actions a – f. Then, the usage of reversal strategies using checkpoints can be seen as a tradeoff between processor run time and memory requirement. To illustrate the checkpointing algorithm resulting from the execution of the actions defined above assume that F equals a stock index forecast for the next eight months the parameters of which have to be adapted. Furthermore suppose that each physical step has a duration of one month. Then for the reversal of F one has to reverse 8 physical steps. Figure 1.2 shows one possible reversal strategy using the actions described above if 3 checkpoints are available. It is assumed that all F_i, $i = 0, \ldots, 7$, the corresponding recording steps $\hat{F}_i$ as well as the reverse

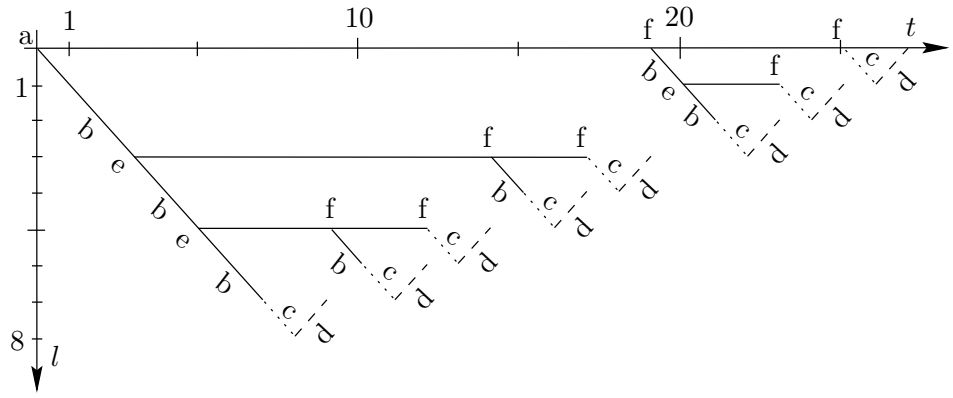


Figure 1.2: Example of Serial Reversal Using 3 Checkpoints

steps $\bar{F}_i$ have the same computational complexity ω. Here and throughout the thesis the physical steps are plotted along the vertical axis and the time required to reverse the execution of F measured in units of ω is represented by the horizontal axis. Hence, the horizontal axis can be thought of as the computational axis. Each solid horizontal line including the computational axis itself represents a checkpoint. The solid slanted lines represent the physical steps F_i without recording whereas the recording steps $\hat{F}_i$ are visualized by dotted slanted lines. Finally the reverse steps $\bar{F}_i$ are drawn as dashed slanted lines. One starts with Action a copying the initial state into a checkpoint, i.e. the computational axis can be seen as a checkpoint. Then Action b is executed by performing three physical steps without recording and state 3 is copied into a checkpoint by Action e. Now again Action b is applied to perform two physical steps and state 5 is copied into a checkpoint (Action e). Subsequently again two physical steps are executed employing Action b a third time. Now Action c and Action d are applied to reverse F_7. After Action f, i.e. after reading state 5 from a checkpoint, a forward sweep starts at state 5 to advance to state 6 (Action b). Hence, it is possible to reverse F_6 by performing Action c and Action d and so on.

In Fig. 1.2 a vertical line intersects for every time t exactly one slanted line and therefore only one processor is applied. Using this checkpointing scheme the time needed for the reversal of F is equal to 27 units ω. Apart from the checkpoints only memory to store one physical step is required. Employing the basic approach the run time would be 16 units ω if all intermediates, i.e. all eight physical steps, were stored as a whole execution log on the tape.

On a multi-processor machine the Actions b, c, d, e, and f that act on different states can be performed at the same time using more than one processor during the reversal process. It is not possible to execute Actions d that reverse different physical steps simultaneously because each reverse step $\bar{F}_{i-1}$ for $i = l, l-1, \ldots, 1$ is based on the results of the previous reverse step $\bar{F}_i$. Nevertheless, using a sufficient number of processors the same run time to calculate the reversal of F as in the basic approach where the complete execution log is recorded can be achieved. In other words, the reversal of F can be performed without any interruption. Figure 1.3 displays one possible implementation of a parallel reversal schedule for the example described above if there are three processors available. This number of processors needed to execute the reversal schedule shown in Fig. 1.3 is given by the maximal number of slanted lines crossing any vertical line. The run time required for the reversal equals 16 units ω and hence is the same as the execution time needed for the basic approach with unrestricted memory. For that the scheme shown in Fig. 1.2 is modified such that the executions of Action b start early enough to enable the completion of the recording steps $\hat{F}_i$ in time.

Naturally, the following question arises in the serial reversal as well as in the parallel reversal: Which states should be copied to checkpoints during the execution of Action e? If more than one processor is used information on when to start a particular execution of Action b – f is required, too. Suitable schemes for reversing the execution of F by determining which state is copied into a particular checkpoint at which time and by fixing the starting times of the

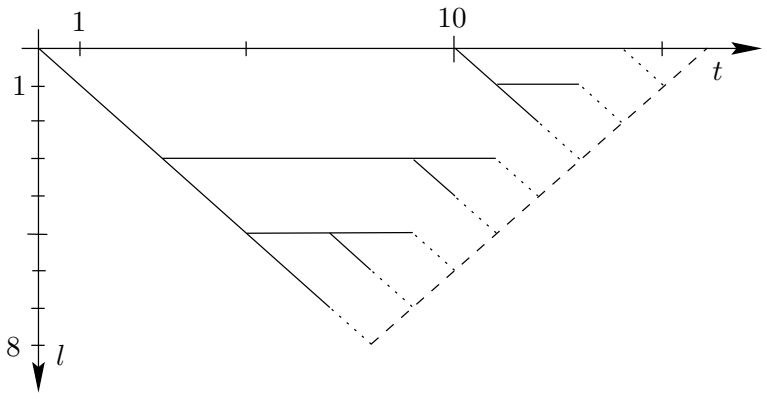


Figure 1.3: Example of Parallel Reversal Using 3 Checkpoints

forward sweeps, recording steps, and the reverse steps in the multi-processor case are called *reversal schedules*.

The subject of this thesis is the construction of reversal schedules that are optimal with respect to particular criteria given certain problem parameters. New reversal schedules for one-processors machines are developed that minimize the temporal complexity for reversing a sequence of physical steps with constant step costs, where the new state depends on $q > 1$ previous states. These evolutionary systems form already a generalization of the one-step case displayed in Fig. 1.1. Then sequences of physical steps with non-constant step costs are considered, where a new state depends only on the previous one. A new technique to find an optimal reversal schedule for these evolutionary systems and serial machines is presented. For parallel machines the maximal number of physical steps that can be reversed with a given number of processors and checkpoints is determined the first time. Corresponding reversal schedules are constructed for multi-processor computers.

The thesis is organized as follows. In Chapter 2 various kinds of evolutionary systems F are introduced. Furthermore several notations are defined and some basic observations are presented.

Chapter 3 deals with reversal techniques in the serial case, i.e. if only one processor is available. In the literature there are several articles concerning optimal reversal schedules if all F_i have the same computational complexity and the new state depends on the previous one only (see e.g. [Gri92], [GPRS96]). Merely one approach is known for a particular two-step method, where the two previous states determine the next one [Cha98]. This thesis extends the existing results in two directions. First, instead of an one-step method as displayed in Fig. 1.1 the evolutionary system F could employ a multi-step method. Therefore optimal reversal schedules for multi-step methods, where the new state depends on $q > 1$ previous states, are constructed theoretically and coded. The second extension concerns the step costs. One-step methods and physical steps with varying computational complexity are considered. A new search algorithm for determining an optimal reversal schedule is represented. For all examples

considered the temporal complexity of the search can be reduced to a quadratic behavior in the number of physical steps to be reversed.

The structure of Chapter 3 is the following: It starts with a presentation of the methods known so far and the software available to reverse the execution of the evaluation procedure F. Then the notations needed are introduced and some basic assertions are proven. After that multi-step methods are considered for the case that all physical steps have the same computational complexity. Two optimality statements and their proofs constructively yielding optimal reversal schedules are presented. The corresponding software is applied to a discretized ordinary differential equation in order to approximate the solution of the adjoint differential equation using various multi-step methods. The run times achieved are reported and interpreted. Then physical steps with varying computational complexity are studied. One has to apply a search algorithm to find an optimal reversal schedule. A particular monotonicity property of the checkpoints is proven the first time. This property is exploited in the search algorithm for an optimal reversal schedule. The improvement of the modified algorithm is reported with the help of several examples.

Chapter 4 covers reversal techniques for multi-processor machines. In an introduction the results of the only published article [Ben96] dealing with parallel reversal schedules are briefly described. Furthermore this thesis presents for the first time a constructive proof for the maximal number of physical steps that can be reversed in minimal time with a given number of processors and checkpoints. This result is derived in two parts. First an upper bound of the number of physical steps that can be reverse with a given number of processors and checkpoints will be established. Then it will be shown how this upper bound can actually be attained. Finally some derived results are shown and several conclusions are drawn.

Chapter 5 presents some conclusions. Implementation challenges with respect to coding the parallel reversal schedules are described. Furthermore conjectures concerning theoretical questions of interest are formulated.

Chapter 2

Notations, Assumptions, and Basic Observations

As mentioned in the previous chapter it is assumed for the development of reversal strategies throughout that the calculation of the evolutionary system F can be split into a sequence of physical steps F_i, $i = 0, \ldots, l-1$. Besides the introduction of several notations this chapter defines different kinds of evolutionary systems F and of reversal schedules S. Furthermore, some basic assumptions and observations are presented.

2.1 One-step Recurrences

In what will be called *one-step recurrence* if a state depends only on its immediate predecessor. Therefore it is possible to describe the calculation of F as the evaluation of the function sequence

$$F_i : \mathbb{R}^{n_i} \;\to\; \mathbb{R}^{n_{i+1}}, \quad \tilde{x}_i \mapsto \tilde{x}_{i+1} \;\equiv\; F_i(\tilde{x}_i), \qquad \text{for} \quad 0 \le i < l \;.$$

All checkpoints are assumed to be of the same size. Hence it is useful to define a state space $\mathbb{R}^M$ with

$$M \;\equiv\; \max\{n_i \mid 0 \le i \le l\} \;.$$

Then it is possible to consider all physical steps F_i as acting on $\mathbb{R}^M$ instead of $\mathbb{R}^{n_i}$. Hence, one has

$$F_i : \mathbb{R}^M \;\to\; \mathbb{R}^M, \quad x_i \mapsto x_{i+1} \;\equiv\; F_i(x_i), \qquad \text{for} \quad 0 \le i < l \;,$$

where some of the F_i do not actually depend on the full state vector x_i but only on a part of it. Using this simplification for one-step recurrences each checkpoint has to provide enough memory to store a complete vector $x \in \mathbb{R}^M$.

For example each state vector x_i may represent a phase state, i.e.

$$x_i \;\equiv\; (y_{i-1}, z_i) \;\equiv\; (f_{i-1}(z_{i-1}), z_i) \;,$$

such that y_{i-1} is determined by the value of a nonlinear function f_{i-1} at the argument z_{i-1} and z_i depends linearly on z_{i-1} and y_{i-1} for $0 < i \le l$. Then

F may serve to approximate the solution of an ordinary differential equation. One has $f_i = f$ for $0 \leq i < l$ if the ordinary differential equation is autonomous.

In the next chapters the counter i is identified with the state x_i to simplify the notation. Moreover, the corresponding evolutionary system F is called *one-step evolution*.

It will be supposed that each physical step F_i applied to x_i has a corresponding reverse step

$$\bar{F}_i : \mathbb{R}^{2\times M} \ \rightarrow \ \mathbb{R}^M, \quad (\bar{x}_{i+1}, x_i) \mapsto \bar{x}_i \ \equiv \ \bar{F}_i(\bar{x}_{i+1}, x_i), \qquad \text{for} \quad l > i \geq 0 \ .$$

For example the $\bar{F}_i$ may be the adjoint of F_i. Then one has

$$\bar{F}_i(\bar{x}_{i+1}, x_i) \ \equiv \ \bar{x}_{i+1}^T F_i'(x_i) \ .$$

with $F_i' = \partial F_i / \partial x_i$ denoting the Jacobian. The sequence $\bar{F}_{l-1}, \ldots, \bar{F}_0$ provides the possibility to calculate the desired value $\bar{x}_0$, i.e. the value of the reversal, for a fixed terminal $\bar{x}_l \in \mathbb{R}^M$. In order to prepare for the corresponding reverse step for each physical step F_i one needs the recording step

$$\hat{F}_i : \mathbb{R}^M \ \rightarrow \ \mathbb{R}^M, \quad x_i \mapsto x_{i+1} \ \equiv \ \hat{F}_i(x_i), \qquad \text{for} \quad 0 \leq i < l \ .$$

It causes the storage of all intermediates calculated during the evaluation of F_i and needed on the way back, i.e. for the execution of the reverse step $\bar{F}_i$.

There exist numerous one-step recurrences (see [HNW96]). They are widely used e.g. for evolution calculations over a time period. The simulation of multibody systems, fluid flows or of the weather represent only a small part of the applications. Using an one-step recurrence in the following example is only one possibility among a great variety of problems to solve and methods to employ.

Example 2.1 *(Explicit Euler Method).* Consider the ordinary differential equation

$$\frac{dz}{dt} = f(z,t) \in \mathbb{R}^n, \quad z(0) = u \in \mathbb{R}^n,$$

with a continously differentiable function $f : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ such that there exists a unique solution $z(t)$. Assume that the evaluation of the evolutionary system F serves to calculate an approximation of $z(\tilde{T})$ for some arbitrary $\tilde{T} \in \mathbb{R}$, $\tilde{T} > 0$. For that one may choose $l \in \mathbb{N}$ and define the physical steps as

$$\begin{aligned} x_{i+1} \ = \ (y_i, z_{i+1}) \ \equiv \ \big(f(z_i, \tilde{t}_i), z_i + h y_i\big) \ \equiv \ F_i(x_i) \quad & \text{with} \\ h \equiv \tilde{T}/l, \ \ \tilde{t}_i \equiv ih, \quad & \text{and} \quad z_i \equiv z(\tilde{t}_i) \end{aligned}$$

for $i = 0, \ldots, l-1$, $\tilde{t}_l \equiv \tilde{T}$, and $z_l \equiv z(\tilde{t}_l)$. It is assumed as throughout that the components of F_i are evaluated from the left to the right, i.e. one computes successively

$$\begin{aligned} y_i \ &= \ f(z_i) \\ z_{i+1} \ &= \ z_i + h y_i \ . \end{aligned}$$

This discretization scheme was first introduced by Euler in the last section of his "Institutiones Calculi Integralis" [Eul68] and uses only the value of f at z_i to

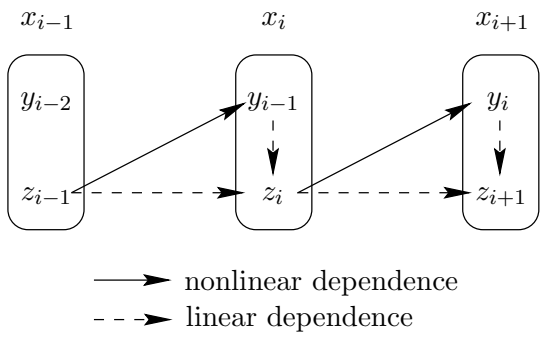


Figure 2.1: Explicit Euler Method

calculate z_{i+1}. Therefore it is called explicit Euler method. The dependencies of the explicit Euler method are illustrated by Fig. 2.1. It is possible to prove that the limit of z_l equals $z(\tilde{T})$ as l approaches infinity [HNW96]. Hence if l is chosen large enough one obtains that the calculated z_l represents a good approximation of $z(\tilde{T})$.

For this particular one-step recurrence $\hat{F}_i$ would evaluate $F_i(x_i)$ storing the intermediate values required on the tape. Then the reverse step $\bar{F}_i$ is given by

$$\bar{x}_i \quad = \quad (\bar{y}_{i-1}, \bar{z}_i) = \left(0, \bar{z}_{i+1} + h\bar{z}_{i+1}^T f'(z_i, \tilde{t}_i)\right)$$

for $i = l-1, \ldots, 0$, where $f'(z_i, \tilde{t}_i)$ denotes the derivative $\frac{\partial f}{\partial z}(z_i, \tilde{t}_i)$. Alternatively, one may write

$$\bar{z}_i \quad = \quad \bar{z}_{i+1} + \bar{y}_i^T f'(z_i, \tilde{t}_i) \; .$$

Because of the phase-state formulation the value $\bar{y}_i = h\bar{z}_{i+1}$ is evaluated internally by $\bar{F}_i$. The adjoint discretization scheme can be derived according to the procedure described in [Gri00]. Hence, the reverse steps $\bar{F}_i$ resemble in some sense the implicit Euler method because the value of f' is needed at state z_i to calculate $\bar{z}_i$.

In the last example the reverse steps represent the adjoint of the discretization scheme. This will always be the case, if the reverse steps $\bar{F}_i$ form the adjoints of the physical steps F_i describing a discretization. A different approach would be to consider the discretized adjoint differential equation as the reversal process. For most discretization schemes both techniques will lead to almost the same results (see e.g. the numerical example of Subsection 3.3.1). Therefore throughout this thesis the adjoint of the discretization scheme will be utilized. Also tools that generate the reverse steps automatically, for example ADOL-C [GJU96], use the adjoint of the discretization scheme instead of the discretized adjoint differential equation.

2.2 Multi-step Recurrences

As an alternative to one-step recurrences, which represent more or less the simplest case of an evolutionary system, it is possible to consider *multi-step*

recurrences, where each new state depends on several previous states.

Suppose that the given evolutionary system F represents a q-step recurrence (see e.g. [HNW96]). It is assumed throughout this thesis that the first q states $x_{1-q}, x_{2-q}, \ldots, x_0$ are known. Usually, they are provided by a start-up calculation. Obviously, one must reverse also the start-up calculation. This has to be done only once. Therefore, the reversal of the start-up calculation is not considered in this thesis. Since the first q states are available F may apply the q-step recurrence in an arbitrary number of physical steps

$$F_i : \mathbb{R}^{q\times M} \to \mathbb{R}^M, \qquad (x_{i-q+1}, \ldots, x_i) \mapsto x_{i+1} \;\equiv\; F_i(x_{i-q+1}, \ldots, x_i),$$

for $0 \le i < l$. Possibly as above some of the F_i do not really depend on all state vectors $x_{i-q+1}, \ldots, x_i$ but only on a part of them.

As in the one-step case each state vector may represent a phase state, i.e.

$$x_i \;\equiv\; (y_{i-1}, z_i) \;=\; (f_{i-1}(z_{i-1}), z_i)\,,$$

such that y_{i-1} is determined by the value of a nonlinear function f_{i-1} at the argument z_{i-1} and z_i depends only linearly on z_{i-j} and y_{i-j} for $1 \le j \le q$ and $0 < i \le l$. Then the evaluation of two physical steps F_i and $F_{\tilde{\imath}}$ do not involve common subexpressions if $i \ne \tilde{\imath}$. This is the reason why y_{i-1} is introduced as part of the ith state vector. The corresponding F may be used to calculate an approximation of the solution of an ordinary differential equation.

In the following chapters the counter i is again identified with the state x_i to simplify the notation. Furthermore, the given evolutionary system F that equals a multi-step recurrence as described above is called *multi-step evolution*. Correspondingly, if a q-step recurrence is employed by F then F is called *q-step evolution*.

As before it is assumed that for each physical step F_i at a certain argument there exists a corresponding reverse step

$$\bar{F}_i : \mathbb{R}^{(q+1)\times M} \to \mathbb{R}^{q\times M}, \quad (\bar{x}_{i-q+1}, \ldots, \bar{x}_i) \mathrel{+}= \bar{F}_i(\bar{x}_{i+1}, x_{i-q+1}, \ldots, x_i),$$

for $l > i \ge 0$. At the beginning of the reversal process all $\bar{x}_i$, $l > i \ge 0$, are initialized to zero. The incremental form is necessary in contrast to the one-step scenario because the x_j occur now as argument to several F_i. Once more the reverse step $\bar{F}_i$ may calculate for example the adjoint of F_i. Then one obtains

$$\bar{F}_i(\bar{x}_{i+1}, x_{i-q+1}, \ldots, x_i) \;\equiv\; \bar{x}_{i+1}^T \left(\frac{\partial F_i}{\partial x_{i-q+j}}(x_{i-q+1}, \ldots, x_i) \right)_{j=1,\ldots,q}. \quad (2.1)$$

In order to execute a reverse step $\bar{F}_i$ one has to ensure that all actions of a call to F_i at a particular argument are recorded on the tape such that the information needed for the reverse step becomes available. To this end the recording steps

$$\hat{F}_i : \mathbb{R}^{q\times M} \to \mathbb{R}^M, \quad (x_{i-q+1}, \ldots, x_i) \mapsto x_{i+1} \;\equiv\; \hat{F}_i(x_{i-q+1}, \ldots, x_i),$$

for $0 \le i < l$ are defined writing all data required to perform $\bar{F}_i$ on the tape.

With respect to the development of reversal strategies a given multi-step recurrence determines the following parameter:

Definition 2.1 (Degree of Nonlinear Dependence). Suppose the given q-step evolution F comprises l physical steps. Let the reverse steps $\bar{F}_i$ serve to calculate the adjoint of F. There exists $\tilde{q} \in \mathbb{N}$ with $0 \leq \tilde{q} \leq q$ such that

$$\frac{\partial F_i}{\partial x_j} = constant$$

for $i - q + 1 \leq j < i - \tilde{q} + 1$ and all $i = 0, \ldots, l - 1$. The integer $\tilde{q}$ is called *degree of nonlinear dependence*. The difference $b \equiv q - \tilde{q}$ denotes the *number of linear arguments* of each physical step F_i.

The number b of linear arguments can be employed for the reversal of multi-step evolutions. For a q-step evolution the corresponding reverse steps $\bar{F}_i$ are defined as in Equation (2.1). The degree of nonlinear dependence yields that the knowledge of the $\tilde{q}$ states $x_{i-\tilde{q}+1}, \ldots, x_i$ suffice to evaluate $\bar{F}_i$, i.e. $\bar{F}_i$ can be reduced to

$$\bar{F}_i : \mathbb{R}^{(\tilde{q}+1)\times N} \to \mathbb{R}^{q\times N}, \quad (\bar{x}_{i-q+1}, \ldots, \bar{x}_i) \mathrel{+}= \bar{F}_i(\bar{x}_{i+1}, x_{i-\tilde{q}+1}, \ldots, x_i) \; .$$

In general b equals the number of reverse steps that can be performed if q successive state vectors $x_{i-q+1}, \ldots, x_i$ and $\bar{x}_{i+1}$ are completely known. In order to reverse the b additional reverse steps one has to perform recording steps that store the intermediates that are needed to perform the reverse steps $\bar{F}_i, \ldots, \bar{F}_{i-b+1}$ on the tape. These recording steps are not identical with $\hat{F}_{i-b+1}, \ldots, \hat{F}_i$. It could happen that it is impossible to evaluate the complete physical step because less than q previous state vectors are known. Nevertheless one can record the data required on the tape. These recording steps have almost the same temporal complexity as the recording steps $\hat{F}_{i-b+1}, \ldots, \hat{F}_i$ and are identified with them. This will be illustrated in the following example. As in the case of one-step evolutions there is a large variety of multi-step evolutions F. Hence in order to illustrate the principle one can consider:

Example 2.2 *(Leap-frog Method).* Let $z(t) \in \mathbb{R}^n$ be the solution of the differential equation

$$\frac{dz}{dt} = f(z) \in \mathbb{R}^n$$

with the initial condition $z_{-1} \equiv z(0) = u \in \mathbb{R}^n$, a continously differentiable function f and a given time interval $[0, \tilde{T}\,]$.

In order to determine the desired value of $z(\tilde{T})$ by the evolutionary system F, the explicit Euler method may serve as start-up calculation to provide the state vector $x_0 = (f(z_{-1}), z_0)$ using the initial value z_{-1}. Then one can use the leap-frog method to calculate an approximation of $z(\tilde{T})$. This yields for a fixed $l \in \mathbb{N}$ with $h = \tilde{T}/(l+2)$

$$x_{i+1} \;=\; (y_i, z_{i+1}) \;\equiv\; (f(z_i), z_{i-1} + 2hy_i) \;\equiv\; F_i(x_{i-1}, x_i) \quad \text{with}$$
$$z_i \equiv z((i+2)h) \quad \text{for} \quad 0 \leq i < l,$$

where the components of F_i are evaluated from the left to the right:

$$\begin{aligned} y_i &= f(z_i) \\ z_{i+1} &= z_{i-1} + 2hy_i \; . \end{aligned}$$

This 2-step method is also known as the midpoint rule (or first Gauss formula). Furthermore, the explicit Nyström method for $k = 1$ proposed by Nyström in 1925 is identical to the leap-frog method [HNW96]. The dependencies of this discretization scheme are illustrated by Fig. 2.2.

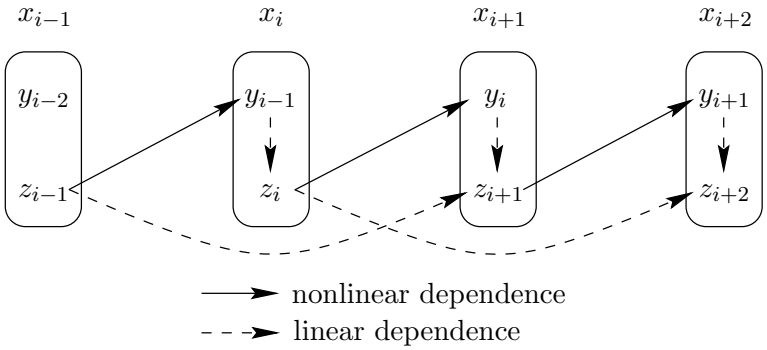


Figure 2.2: Leap Frog Method

Now suppose the reverse steps $\bar{F}_i$ are equal to the adjoint of F_i. One obtains for the derivatives of F_i

$$\begin{aligned} \frac{\partial F_i}{\partial y_{i-2}}(x_{i-1}, x_i) &= (0,0)^T, & \frac{\partial F_i}{\partial z_{i-1}}(x_{i-1}, x_i) &= (0, I)^T \\ \frac{\partial F_i}{\partial y_{i-1}}(x_{i-1}, x_i) &= (0,0)^T, & \frac{\partial F_i}{\partial z_i}(x_{i-1}, x_i) &= (f'(z_i), 2hf'(z_i))^T \end{aligned}$$

and hence

$$\frac{\partial F_i}{\partial x_{i-1}}(x_{i-1}, x_i) = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} \quad \text{and} \quad \frac{\partial F_i}{\partial x_i}(x_{i-1}, x_i) = \begin{pmatrix} 0 & f'(z_i) \\ 0 & 2hf'(z_i) \end{pmatrix} . \tag{2.2}$$

Therefore one finds according to Equation (2.1) or the procedure described in [Gri00] with $\bar{x}_l \equiv (0, \bar{z}_l)^T$

$$(\bar{x}_{i-1}, \bar{x}_i) \;=\; \big((\bar{y}_{i-2}, \bar{z}_{i-1}), (\bar{y}_{i-1}, \bar{z}_i)\big) \mathrel{+}= \; \big((0, \bar{z}_{i+1}), (0, 2h\bar{z}_{i+1}^T f'(z_i))\big)$$

for $l > i \geq 0$ and therefore

$$\bar{F}_i(\bar{x}_{i+1}, x_i) \;\equiv\; \big((0, \bar{z}_{i+1}), (0, 2h\bar{z}_{i+1}^T f'(z_i))\big) .$$

In order to perform these reverse steps the recording step $\hat{F}_i$ has to store the intermediates calculated during the evaluation of f at the argument z_i on the tape besides the evaluation of F_i.

From Equation (2.2) follows according to Def. 2.1 that the degree of nonlinear dependence is equal to 1. Hence one obtains $b = 1$. To illustrate the meaning of the number b of linear arguments suppose that $\bar{x}_i, x_{i-1}$, and x_i are known. Then one can perform a recording step that stores the intermediates calculated during the evaluation of f at the argument z_{i-1} on the tape. Therefore it is possible to evaluate $\bar{F}_{i-1}$ since all needed values are available. It follows

again that b is equal to 1 for the leap-frog method. In comparison the recording step $\hat{F}_{i-1}$ would store the linear dependencies besides the intermediates calculated during the evaluation of f at the argument z_{i-1} on the tape. Therefore $\hat{F}_{i-1}$ and the recording step performed without the complete evaluation of the physical step have almost the same temporal complexity.

2.3 Characterizing the Step Costs

Throughout the thesis the symbols F_i, $\hat{F}_i$, and $\bar{F}_i$ are used to denote both mathematical functions and procedures to evaluate the corresponding mathematical function on a computer. The former interpretation applies when arguments are explicitly listed. Hence, one has $F_i(x) = \hat{F}_i(x)$ for all arguments x, i.e. advancing and recording are equivalent as functions. Considered as procedures $\hat{F}_i$ has the side effect of recording the data required for the evaluation of the reverse step $\bar{F}_i$ on the tape.

The spatial complexity measure $|F_i|$ counts for the physical step F_i with $0 \leq i < l$ the number of bytes one has to store on the tape to prepare the execution of the reverse step $\bar{F}_i$. For the development of the reversal schedules in the next chapters the physical steps are assumed to be chosen such that the memory requirement for recording one of them, i.e. $|F_i|$, is greater than the space needed to store one state vector. Hence, the spatial complexity should be greater than the number of bytes needed to store N floating point numbers. Furthermore, it is required that an upper bound of $|F_i|$, $i = 0, \ldots, l-1$, is available at a negligible cost. Moreover, the spatial complexity $|F_i|$ should be essentially constant.

It is assumed throughout that one has

$$TIME\big(\text{evaluate } F_0, \ldots, F_{l-1}\big) \;=\; \sum_{i=0}^{l-1} TIME\big(\text{evaluate } F_i\big),$$

i.e. the physical steps are separable, which means that there are no common subexpressions involved in the evaluations of various physical steps. Then the temporal complexity of F_i, $0 \leq i < l$, i.e. the time needed to execute a particular physical step, will be denoted throughout as

$$t_i \;\equiv\; TIME(F_i) \quad \text{for} \quad 0 \leq i < l\,.$$

This definition is only appropriate if the physical steps are separable as supposed in this thesis. The time needed to execute $\hat{F}_i$, $0 \leq i < l$, will be denoted from now on as

$$\hat{t}_i \;\equiv\; TIME\big(\hat{F}_i\big) \quad \text{for} \quad 0 \leq i < l\,.$$

Moreover throughout the temporal complexity of $\bar{F}_i$, $l > i \geq 0$, i.e. the time needed to perform a particular reverse step, will be denoted as

$$\bar{t}_i \;\equiv\; TIME\big(\bar{F}_i\big) \quad \text{for} \quad l > i \geq 0\,.$$

For example all linear multi-step methods to approximate the solution of a differential equation can be formulated according to the definition of multi-step recurrence used in this thesis. The physical steps are separable if the discretization utilizes phase states as described above. Furthermore, also the backward differentiation formulas (or BDF-methods, see e.g. [Gea71]) that involve an inverse function fit the model employed.

Having the definitions of the temporal complexity of each physical step and of the corresponding recording and reverse steps at hand one can make a further distinction between different one-step recurrences and multi-step recurrences, respectively:

Definition 2.2 (Uniform step costs). Suppose the given evolutionary system F is formed by the physical steps F_i with $0 \leq i < l$. If there exists a constant $\omega \in \mathbb{R}$, such that $t_i = \omega$ is valid for all F_i, $0 \leq i < l$, the physical steps have *uniform* step costs. Otherwise the physical steps F_i, $0 \leq i < l$, have *non-uniform* step costs.

The following evolutionary systems may serve as illustration for the two different kinds of step costs considered in this thesis.

Example 2.3 *(Explicit Situation).* For each physical step using the explicit Euler method introduced in Example 2.1 or the leap-frog method as in Example 2.2 one evaluation of $f(x,t)$ and $f(x)$, respectively, becomes necessary. If the calculations of $f(x,t)$ and $f(x)$ cause the same temporal complexity for each possible argument, as it is usually the case, the corresponding physical steps F_i, $0 \leq i < l$, have uniform step costs.

Example 2.4 *(Implicit Situation).* Assume that the evolutionary system F consists of physical steps each of which is defined implicitly so that an iterative process has to be performed during the evaluation of each physical step. If the duration of this iteration depends strongly on the argument actually considered it is likely that the physical steps have non-uniform step costs.

2.4 Reversal Schedules

In order to perform the reversal of a given evolutionary system F one has to evaluate several Actions a, b, c, d, e, and f defined already in the previous chapter. For that purpose one has to distinguish between single- and multiprocessor machines.

Serial Reversal Schedules

Suppose that only one processor is available to perform the reversal process of a given evolutionary system. For describing the strategy employed in more detail one may use the following

Definition 2.3 (Serial Reversal Schedule S). Assume the q-step evolution F under consideration comprises the physical steps F_i, $0 \leq i < l$, and determines $b \leq q$ according to Def. 2.1. Let $n \equiv cq$, $c \in \mathbb{N}$, checkpoints be available

each of which can accommodate one state vector. Then a *serial reversal schedule* S initializes $l_c = l$ and $i = 0$. Subsequently starting with $1 - q, \ldots, i$ it performs a sequence of basic actions

$$\begin{array}{lcl} P_m & \equiv & \text{Increment } i \text{ by } m \in \{1, \ldots, l_c - i - 1\} \\ D & \equiv & \text{If } i \geq l_c - q\text{: Decrement } l_c \text{ to } i \\ W_j & \equiv & \text{Copy } i - q + 1, \ldots, i \text{ to checkpoints } j - q + 1, \ldots, j \\ & & \text{with } j \in \{q, 2q, 3q, \ldots, cq\} \\ R_j & \equiv & \text{Reset } i - q + 1, \ldots, i \text{ to checkpoints } j - q + 1, \ldots, j \\ & & \text{with } j \in \{q, 2q, 3q, \ldots, cq\} \\ A_j & \equiv & \text{If } j = i = l_c - 1\text{: Decrement } l_c \text{ by } b \end{array}$$

until l_c has been reduced to zero, i.e. the reversal of F is finished.

This definition of a reversal schedule relies closely on the one described in Chapter 12 of [Gri00]. Furthermore, it is related to the execution of the actions defined in Chapter 1. Hence P_m corresponds to the forward sweep of Action b, i.e. the physical steps $F_h, h = i, \ldots, i + m - 1$, are performed. Action c and d are combined to D in order to reverse the final physical step(s) by evaluating $\hat{F}_i, \ldots, \hat{F}_{l_c-1}$ and $\bar{F}_{l_c-1}, \ldots, \bar{F}_i$. The checkpoint writing of Action e is done by W_j. Action f, i.e. the checkpoint reading, is identical to R_j. The basic action A_j is only relevant for the multi-step evolutions considered in this thesis for the first time with respect to reversal schedules. Hence A_j does not occur in the reversal strategies that use checkpoints up to now. As can be seen it is supposed that q state vectors are held by the processor in order to perform for example the physical steps.

It is always possible to find a serial reversal schedule for a given q-step evolution F if at least q checkpoints can accommodate q state vectors as one may use the serial reversal schedule S determined by the sequence

$$\begin{array}{rcl} S & \equiv & W_1 + P_{l-1} + D + R_1 + P_{l-2} + D + R_1 + \cdots + P_k + D + R_1 + \\ & & \cdots + P_1 + D + R_1 + D, \end{array}$$

where $+$ indicates successive performance. This reversal schedule represents the strategy of total recalculation described in the next chapter.

Throughout the thesis it is supposed that the following assumption holds:

Assumption 2.1 (Available Memory). For a given q-step evolution F combining l physical steps let $n \equiv cq$ with $c \in \mathbb{N}$ be valid. The number of checkpoints that are available equals n. Moreover one has always the possibility to store the writings of q recording steps onto the tape.

Therefore it makes sense to say that a serial reversal schedule needs c checkpoints each of which combines q state vectors.

As can be seen from the definition of serial reversal schedules for each physical step F_i, $i = 0, \ldots, l - 1$, of the evaluation procedure F one has to perform both the recording step $\hat{F}_i$ and the reverse step $\bar{F}_i$ exactly once, respectively, during the reversal of F. Hence the time needed by any serial reversal schedule to record the data on the tape is given by $\sum_{i=0}^{l-1} \hat{t}_i$. Equivalently the time needed

by any serial reversal schedule to perform the reverse steps $\bar{F}_i$ equals $\sum_{i=0}^{l-1} \bar{t}_i$. Furthermore, the temporal complexity of the basic action D and a particular value of i is given by $\hat{t}_i + \cdots + \hat{t}_{l_c-1} + \bar{t}_{l_c-1} + \cdots + \bar{t}_i$. In the figures used throughout this thesis to illustrate the reversal schedules this sum corresponds to the time needed to perform the "hook" consisting of a dotted and a dashed line for the particular value of i (see e.g. Fig. 1.2). With respect to the physical steps that are performed it is helpful to have the following definition at hand.

Definition 2.4 (Executions of Physical Steps). Suppose the given evolutionary system F comprises l physical steps. A reversal schedule S for F determines for $i = 0, \ldots, l-1$ the integer counts

$$r_i \quad \equiv \quad \text{number of times the physical step } F_i \text{ is performed for executing } S \, .$$

Chapter 3 examines the optimization of the temporal complexity caused by a serial reversal schedule S, i.e. the sum

$$\sum_{i=0}^{l-1} r_i t_i \, .$$

For uniform step costs $t_i = \omega$, $0 \leq i < l$, this sum can be reduced to $\omega \sum_{i=0}^{l-1} r_i$, which measures the number of physical steps performed during the reversal. In order to find a corresponding optimal serial reversal schedule for uniform as well as for non-uniform step costs dynamic programming techniques are used.

Parallel Reversal Schedules

Obviously serial reversal schedules can be used on single- as well as on multi-processor machines to perform the reversal of an evolutionary system F consisting of the appropriate number of physical steps. Naturally, one can use more than one processor for the reversal of a given evolutionary system F. For that it is assumed for simplicity that all processors work on a common memory, i.e. the shared memory model is used. Nevertheless, the parallel reversal schedules can also be implemented using a message passing strategy. Here one has to take the time needed for the communication between the processors into account. Using the parallel reversal schedules considered in this thesis the communication cost is largely determined a priori. This thesis constructs reversal schedules on multi-processor machines for the following one-step evolutions:

Definition 2.5 (Uniform One-step Evolutions). Suppose the given one-step evolution F comprises l physical steps with uniform step costs $t_i = \omega \in \mathbb{R}$. If there exist $\hat{\omega}, \bar{\omega} \in \mathbb{N}$ with $\hat{t}_i = \hat{\omega} t_i$ and $\bar{t}_i = \bar{\omega} t_i$ then F is called *uniform one-step evolution*.

Obviously, the step costs of uniform one-step evolutions can be normalized to $t_i = 1$, $\hat{t}_i = \hat{\omega} \in \mathbb{N}$, and $\bar{t}_i = \bar{\omega} \in \mathbb{N}$. Then the usage of two-dimensional arrays establishes one possibility to describe the reversal process of an uniform one-step evolution:

Definition 2.6 (Reversal Trace). Suppose the uniform one-step evolution F under consideration comprises l physical steps. Let c be the number of available checkpoints, p the number of available processors, and $k = p + c$. The given two-dimensional array T has entries $T[j][h] \in \{\imath, \vec{\imath}, \hat{\imath}, \bar{\imath} \mid 0 \leq \imath < l\}$ for $0 \leq j \leq N$ with $N \in \mathbb{N}$ and $1 \leq h \leq k$. Define $T[j] \equiv \{T[j][1], \ldots, T[j][k]\}$ for $0 \leq j \leq N$. If T fulfils the requirements

1. Initial configuration: $T[0][h] = 0$, $1 \leq h \leq k$
2. Terminal configuration: $T[N][h] = 0$, $1 \leq h \leq k$
3. Advances: $\vec{\imath} \in T[j] \Rightarrow \jmath \in T[j-1]$ or $\vec{\jmath} \in T[j-1]$, $\jmath = \max\{\imath - 1, 0\}$
4. Decrements: For each $\imath \in \{0, \ldots, l-1\}$ there exists $j_\imath \in \{\hat{t}, \ldots, N - \bar{t} - 1\}$, a recording step $\hat{\imath} \in T[j], j_\imath - \hat{t} + 1 \leq j \leq j_\imath$ such that $\jmath \in T[j_\imath - \hat{t}\,]$ or $\vec{\jmath} \in T[j_\imath - \hat{t}\,], \jmath = \max\{\imath - 1, 0\}$, and a reverse step $\bar{\imath} \in T[j], j_\imath < j \leq j_\imath + \bar{t}$
5. Fitting reversals: $j_{\imath-1} < j_\imath - \bar{t} + 1$ for $\imath \in \{1, \ldots, l-1\}$
6. Admissibility: The number of processors used in $T[j]$ is not greater than p for $1 \leq j \leq N$.

then T is called *reversal trace*.

An entry $\imath$ of T stands for a checkpoint storing state $\imath$. The value $\vec{\imath}$ represents the evaluation of the physical step $F_\imath$. The execution of the recording step $\hat{F}_\imath$ is denoted by $\hat{\imath}$. Finally $\bar{\imath}$ is identified with $\bar{F}_\imath$.

Each reversal trace represents at least one reversal schedule on a multiprocessor machine. The next example will illustrate this meaning of a reversal trace.

Example 2.5 *(Reversal Trace and Parallel Reversal Schedule).* Let the uniform one-step evolution F to be reversed combine 5 physical steps. Suppose F determines the temporal complexities $t_i = 1$, $\hat{t}_i = 3$, and $\bar{t}_i = 2$ for $i = 0, \ldots, 4$. One checkpoint and three processors are available. Figure 2.3 illustrates one possible reversal of F. The only task left is to determine which processor has to perform which action. For example the first processor could perform the physical steps

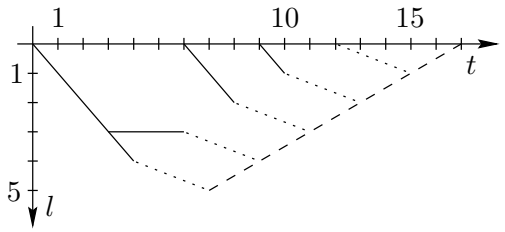


Figure 2.3: Possible Reversal of F

$F_0, \ldots, F_3$, the recording step $\hat{F}_4$, and all reverse steps $\bar{F}_i$, $4 \geq i \geq 0$. The second processor could be used as checkpoint storing state 3. Subsequently it

could evaluate the sequence $\hat{F}_3$, F_0, and $\hat{F}_1$ in time. The physical steps F_0, F_1, and the recording steps $\hat{F}_2$ and $\hat{F}_0$ could be performed by the third processor.

Furthermore, Fig. 2.3 displays the two-dimensional array $T[j][h]$, $0 \leq j \leq 18$ and $1 \leq h \leq 4$, the entries of which are defined as in Table 2.1. As can be seen, T fulfils the requirements $1-6$ of Def. 2.6. Hence T represents a reversal trace.

$h \backslash j$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	$\vec{0}$	$\vec{1}$	$\hat{2}$	$\vec{0}$	$\hat{1}$	0	$\hat{0}$	0	0	0	0	0
3	0	0	0	0	3	3	3	$\hat{3}$	$\hat{3}$	$\hat{3}$	$\hat{2}$	$\hat{2}$	$\hat{1}$	$\hat{1}$	$\hat{0}$	$\hat{0}$	0	0	0
4	0	$\vec{0}$	$\vec{1}$	$\vec{2}$	$\vec{3}$	$\hat{4}$	$\hat{4}$	$\hat{4}$	$\bar{4}$	$\bar{4}$	$\bar{3}$	$\bar{3}$	$\bar{2}$	$\bar{2}$	$\bar{1}$	$\bar{1}$	$\bar{0}$	$\bar{0}$	0

Table 2.1: Reversal Trace T Corresponding to Fig. 2.3

For each reversal trace T it is possible to construct an appropriate parallel reversal schedule by determining which action $\vec{\imath}, \hat{\imath}, \bar{\imath}$ is executed by which processor and which checkpoint stores which state $\imath$. Usually there exists a variety of reversal schedules corresponding to the reversal trace T. In the last example one obtains easily another parallel reversal schedule for F than the one described above. One possibility could be to use the third processor as checkpoint storing state 3 instead of the second processor. It might be difficult to find an "optimal" parallel reversal schedule that minimizes the communication required for a given reversal trace T. Nevertheless one can define an "equivalence class" of parallel reversal schedules which have the same reversal trace T. These reversal traces determine yet important properties of the "equivalence classes" introduced by them. Therefore in Chapter 4 only the reversal traces, i.e. "equivalence class" of parallel reversal schedules, are considered. They are used to determine the maximal number of physical steps that can be reversed with a given number of processors and checkpoints.

Throughout figures as for example Fig. 2.3 are used to illustrate reversal traces. Here, the number p_j of slanted lines crossing any vertical line in the interval $(j-1, j)$ denotes the number of processors needed to execute $T[j]$. The number of processors p required by T is equal to $\max\{\, p_j \mid 0 \leq j \leq N \,\}$. Furthermore, the number of checkpoints c_j needed for $T[j]$ is given by the number of horizontal lines crossing a vertical line at $t \in (j-1, j)$. Hence $s_j = p_j + c_j$ denotes the number of resources required by $T[j]$.

Chapter 3

Serial Reversal Schedules

3.1 Introduction

This chapter considers the reversal of a given evolutionary system F comprising l physical steps F_i, $0 \leq i < l$, as defined in the previous chapter if only one processor is available.

An obvious way to reverse F is given by recording the complete execution log onto a tape during the evaluation of the physical steps F_i, $0 \leq i < l$, and reading the tape backward during the reversal. I.e. first the recording steps $\hat{F}_i$, $0 \leq i < l$, and second the reverse steps $\bar{F}_i$, $l > i \geq 0$, are evaluated. Figure 3.1 illustrates this basic approach for an evolutionary system F combining 12 physical steps with the temporal complexities $\hat{t}_i = \bar{t}_i = 1$. All intermediate values are recorded onto a tape, a process that is represented by the dotted slanted line. Subsequently the reversal depicted as dashed slanted line is performed without any interruption on the basis of the data stored. This approach leads

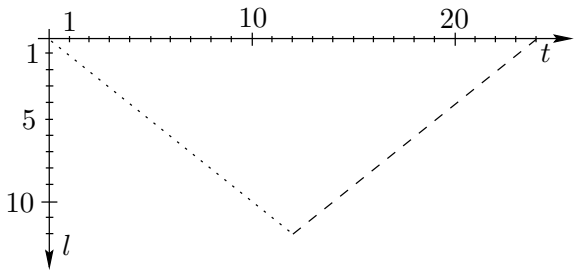


Figure 3.1: Reversal Process Storing All Intermediates

to a run time for the reversal that is only a small multiple of the run time needed to evaluate F, but the spatial complexity is proportional to the temporal complexity of F (see e.g. [Gri89], originally due to Linnainmaa [Lin76]). Hence if F represents a large evaluation with thousands of physical steps each of which contains a reasonable amount of data to record then this technique is not applicable because of the enormous amount of memory required.

At the opposite extreme the spatial complexity is minimized if one recalculates all data needed for the reversal as depicted in Fig. 3.2. Since all physical

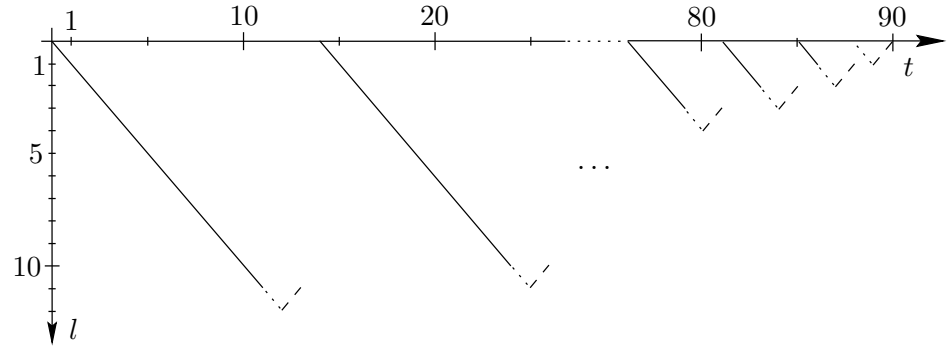


Figure 3.2: Total Recalculation for $l = 12$

steps F_i, $0 \leq i < l$, are evaluated $l - 1 - i$ times, the complete elimination of storage except the checkpoint storing the initial state 0 and one recording step is quite expensive in terms of run time. If all temporal complexities t_i are of roughly the same size the run time required for the reversal would be about $\frac{1}{2}l^2$ times that of one physical step F_i. Hence this approach yields an increase in temporal complexity that is probably unacceptable.

A first idea to reduce the memory requirement at a moderate increase in run time could be to place c checkpoints during the first forward sweep advancing to state $l - 1$. This provides the possibility to start the calculation of F at the checkpoints again. Then each piece of the evaluation determined by the checkpoints set is reversed separately. One possible resulting serial reversal schedule for $l = 12$ using 3 checkpoints that are placed equidistantly is shown in Fig. 3.3. Using this technique the memory requirement depends linearly on

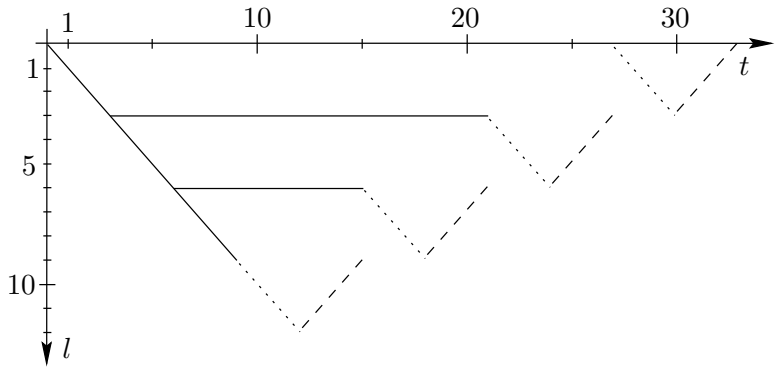


Figure 3.3: Reversal Process with no Reuse of Checkpoints

the number of physical steps l for a fixed number c of available checkpoints. The increase in run time is limited by the run time needed to evaluate F. Here, the main point is that no checkpoint is ever reused. This checkpointing strategy is explained further and implemented in [MO] for the calculation of derivatives using the reverse mode of algorithmic differentiation. If the solution

of a differential equation is approximated using the leap-frog scheme, where q equals 2 (see Chapter 2), a particular reversal technique without reuse of the checkpoints is discussed in [Cha98] in order to approximate the solution of the corresponding adjoint differential equation with a reduced memory requirement.

Nevertheless in the case of one-step evolutions ($q = 1$) with uniform step costs it is possible to utilize a more sophisticated reversal strategy, namely by reusing the checkpoints appropriately. Here, one can achieve a logarithmic behavior of the memory requirement and of the increase in run time if $q = 1$ [Gri92]. Further compromises between temporal and spatial complexity are possible, so that the reversal becomes applicable to evolutionary systems F of virtually any size. Figure 3.4 shows one possibility to reverse 12 physical steps reusing the checkpoints if 4 checkpoints are available. During the reversal of the physical steps $F_0, \ldots, F_4$ once more state vectors are copied to the 3 free checkpoints. All known results with respect to the reversal technique with reuse

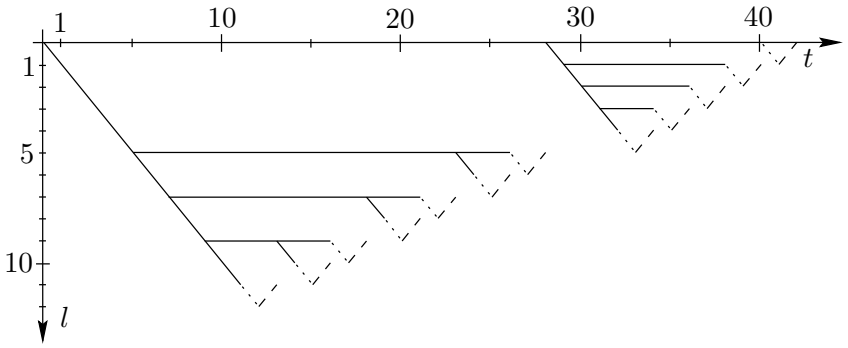


Figure 3.4: Reversal Process Reusing Checkpoints

of checkpoints consider one-step evolutions, i.e. $q = 1$, and are only optimal when the step costs are uniform. This approach was introduced first by [Gri92] showing that a logarithmic growth both in the temporal and spatial complexity can be achieved. An explicit description of the increase in run time can be found in [GPRS96] and using a different proof in [GW00]. Furthermore, the latter proposes serial reversal schedules minimizing also the number of checkpoint writings. Two implementation of this checkpointing strategy for one-step evolutions with uniform step costs are available, namely [GW00] and [Kub98].

These existing results will be extended on one hand to multi-step evolutions and on the other hand with respect to the uniformity of the step costs. Following the introduction of the notations needed and the proofs of some general properties in Section 3.2 two optimality results are shown in Section 3.3. These will lead to the construction of optimal serial reversal schedules for multi-step evolutions with uniform step costs. Run-time results applying this new reversal technique are reported and interpreted for the approximations of the solution of an ordinary differential equation and its adjoint. In Section 3.4 one-step evolutions with non-uniform step costs are considered. Here, a search algorithm has to be applied. For the first time a search algorithm based on dynamic programming is proposed. Using a certain monotonicity property of checkpoint

partitions one can improved this first approach drastically. Several examples are examined and the enormous shortening of the run time needed to determine one optimal serial reversal schedule is shown.

3.2 Notations and Basic Observations

The time needed to perform the recording steps as well as the temporal complexity to perform the reverse steps required is determined a priori (see Chapter 2 for an explanation). Only the time needed to perform physical steps F_i will vary from one serial reversal schedule to the other. Therefore this forms a good criterion for judging a particular serial reversal schedule.

Definition 3.1 (Cost $t_S(i,h,c)$, Minimal Cost $t(i,h,c)$). Suppose a q-step evolution F combining l physical steps and c checkpoints are given. Let S be a serial reversal schedule using no more than c checkpoints to reverse the physical steps F_j between state i and state h with $0 \leq i \leq h \leq l$. The *cost* $t_S(i,h,c)$ to reverse the sequence $F_i, \ldots, F_{h-1}$ applying S is defined as

$$t_S(i,h,c) \equiv \sum_{j=i}^{h-1} r_j t_j \ ,$$

where r_j denotes the number of times the physical step F_j is performed during the reversal applying S as defined in Chapter 2. The *minimal cost* $t(i,h,c)$ for the reversal of the physical steps between state i and state h is denoted as

$$t(i,h,c) \equiv \min\{t_S(i,h,c) \mid S \text{ is a serial reversal schedule to reverse the physical steps between state } i \text{ and state } h\} \ .$$

If the sequence of physical steps under consideration has uniform step costs $t_i = \omega$ only the number of physical steps performed during the execution of a particular serial reversal schedule S is important because one has

$$t_S(i,h,c) = \sum_{j=i}^{h-1} r_j t_j = \omega \sum_{j=i}^{h-1} r_j \ .$$

If the physical steps have non-uniform step costs this is obviously not the case as illustrated by the following

Example 3.1 *(Cost of Serial Reversal Schedules).* Let the one-step evolution F consist of 9 physical steps. Consider the serial reversal schedule

$$S \equiv W_2 + P_3 + W_1 + P_2 + D + R_1 + P_1 + D + R_1 + D + R_2 + P_1 + W_1 + P_1 + D + R_1 + D + R_2 + D,$$

which uses 2 checkpoints for the reversal of 6 physical steps. If the basic action D causes also the release of a checkpoint storing the state $l_c - 1$ Fig. 3.5 illustrates the distribution of the checkpoints and the processor caused by S. Hence one

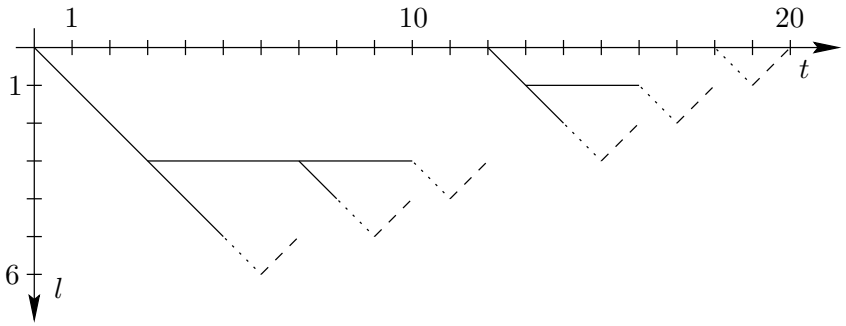


Figure 3.5: Distribution of Checkpoints and Processors determined by S

obtains for the integer counts r_i defined according to Def. 2.4

$$r_5 = 0, \qquad r_2 = r_4 = 1, \qquad r_0 = r_1 = r_3 = 2 \; .$$

If the physical steps have uniform step costs $t_i = 1$, $i = 0, \ldots, 8$, this yields

$$t_S(0, 6, 2) = \sum_{i=0}^{5} r_i = 8 = t_S(3, 9, 2)$$

if S is applied to reverse the physical steps between state 0 and state 6 and between state 3 and state 9, respectively. It can be proven (see e.g. [GW00]) that this equals also the value of $t(0, 6, 2)$ and $t(3, 9, 2)$, respectively.

Now suppose that there is a start-up calculation with a greater temporal complexity comprising the first two physical steps, i.e. $t_i = 3$ for $i = 0, 1$ and $t_i = 1$ for $i = 2, \ldots, 8$. If S is applied the same way it follows that

$$t_S(0, 6, 2) = \sum_{i=0}^{5} r_i \, t_i = 16 \neq 8 = t_S(3, 9, 2) \; .$$

Furthermore, one has $t_S(0, 6, 2) = 16 > 15 = t_{\tilde{S}}(0, 6, 2)$, where $\tilde{S}$ is defined as the serial reversal schedule

$$W_2 + P_2 + W_1 + P_3 + D + R_1 + P_2 + D + R_1 + P_1 + D + R_1 + \\ D + R_2 + P_1 + D + R_2 + D \; .$$

Figure 3.6 displays this serial reversal schedule $\tilde{S}$. Obviously, S does not attain the minimal cost $t(0, 6, 2)$ for the considered one-step evolution with non-uniform step costs.

To derive an explicit formula for $t(i, h, c)$ for multi-step evolutions with uniform step costs and to improve the search algorithm for one-step evolutions with non-uniform step costs the serial reversal schedule will be decomposed into smaller substructures that are considered separately. To that end it is necessary to prove the following assertion.

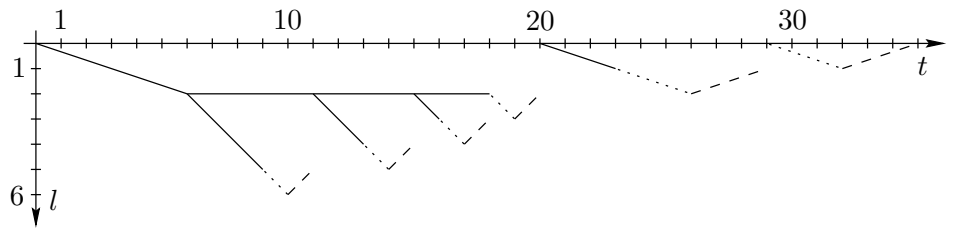


Figure 3.6: Distribution of Checkpoints and Processors determined by $\tilde{S}$

Lemma 3.1 (Checkpoint Persistence).
Suppose the given q-step evolution F consists of l physical steps. If $q > 1$ the physical steps of F must have uniform step costs. Otherwise also non-uniform step costs are allowed. Let S be a serial reversal schedule that uses no more than c checkpoints and attains the minimum cost $t(0, l, c)$. Then S can be modified without increasing the cost $t_S(0, l, c) = t(0, l, c)$ such that after the checkpoint writing W_j copying states $i - q + 1, \ldots, i$ in checkpoint j the next action W_j occurs only when l_c has already been reduced to i or below. Moreover, until that time no actions involving the states between 0 and i are taken.

Proof. Suppose, S is a serial reversal schedule where the action W_j is taken at states $i - q + 1, \ldots, i$ and then later again at some states $\tilde{\imath} - q + 1, \ldots, \tilde{\imath}$ with $l_c > i$ all the time. It is not possible that a forward sweep P_m advances through the states stored in checkpoint j because then P_m could be replaced by $R_j + P_{\tilde{m}}$ with $\tilde{m} < m$ reducing $t_S(0, l, c)$. This is a contradiction to the assumption that the reversal schedule under consideration minimizes $t_S(0, l, c)$.

Hence, all that can happen concerning the states $i - q + 1, \ldots, i$ in between the two successive actions W_j are one forward sweep immediately after the first action W_j or resets R_j followed by forward sweeps over various length.

If no action R_j occurs the first action W_j is useless and can be deleted. The same is true if only one advance P_m is performed right after the first action W_j.

If no forward sweep is performed immediately after the first action W_j only resets R_j followed by forward sweeps over various length may occur.

If there is only one reset R_j possibly followed by an advance consider the actions between the first W_j and the only R_j. The actions, which affect only states between 0 and i, can be performed also after the next action W_j. Furthermore, S can be modified in such a way that all other actions between the first W_j and the only R_j are executed after R_j and the actions belonging to this R_j. Then the first checkpoint writing W_j and the only reset R_j are useless and can be deleted.

If there is more than one reset R_j the overall cost would be reduced if the advance before the first action W_j goes one physical step further and the first W_j is performed copying states $i - q + 2, \ldots, i + 1$ instead of $i - q + 1, \ldots, i$ because the length of all forward sweeps performed after the resets R_j is reduced by one physical step. This is a contradiction to the assumption that the reversal schedule under consideration minimizes $t_S(0, l, c)$.

If an advance is performed right after the first W_j and one or more reset R_j

occur between the two successive actions W_j all advances would be reduced in length if one performs W_j at least one step further such that W_j copies states $i-q+2,\ldots,i+1$ in the first place. This is a contradiction to the assumption that S attains $t(0,l,c)$, which completes the proof. ∎

The concept "Checkpoint persistence" has already been introduced for one-step evolutions in Chapter 12 of [Gri00]. Also the proof of this property relies on the one presented in the same book. Lemma 3.1 implies for any q-step evolution F consisting of l physical steps and determining the parameter b according to Def. 2.1 the following powerful conclusion: An optimal serial reversal schedule S for reversing l physical steps with up to c checkpoints can be decomposed without loss of generality into subschedules dealing with the sequences $F_0,\ldots,F_{\tilde{l}-b-1}$ and $F_{\tilde{l}},\ldots,F_{l-1}$ of physical steps, respectively. After copying the initial states $1-q,\ldots,0$ to a checkpoint a forward sweep to some state $\tilde{l}$ and the action W_j are performed. Then the sequence $F_{\tilde{l}},\ldots,F_{l-1}$ is reversed using the remaining checkpoints. After that the action A_j causes the reverse steps $\bar{F}_{\tilde{l}-1},\ldots,\bar{F}_{\tilde{l}-b}$ during the release of the checkpoint storing the states $\tilde{l}-q+1,\ldots,\tilde{l}$. Finally the reversal of the sequence $F_0,\ldots,F_{\tilde{l}-b-1}$ is performed using again c checkpoints. This situation is depicted by Fig. 3.7. Hence it is possible to prove the following assertion:

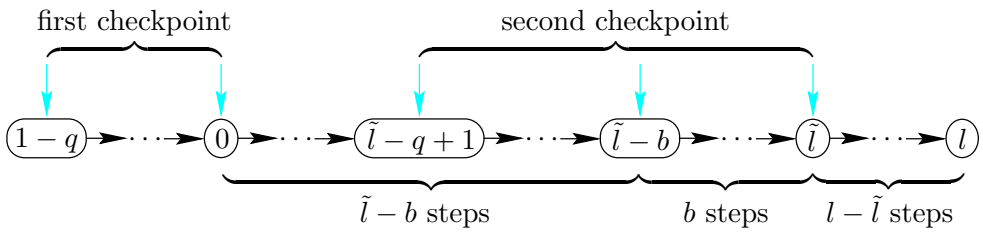


Figure 3.7: Placing of the Second Checkpoint Writing

Lemma 3.2 (Decomposition of Serial Reversal Schedules).
Let F be a q-step evolution that determines b according to Def. 2.1. If $q>1$ the physical steps of F must have uniform step costs. Otherwise also non-uniform step costs are allowed. Then the function $t(i,h,c)$, which denotes the minimal cost to reverse the physical steps F_j between state i and state h of F with up to c checkpoints accommodated at any time, equals 0 if $h-i\le q$. If $h-i>q$ then $t(i,h,c)$ has the form

$$t(i,h,c) = \begin{cases} \displaystyle\sum_{j=i}^{h-1} \left\lfloor \frac{h-j-1}{q} \right\rfloor t_j & \text{for } c=1, \\ \displaystyle\min_{i+q\le\tilde{l}<h} \left\{ \sum_{j=i}^{\tilde{l}-1} t_j + t(\tilde{l},h,c-1) + t(i,\tilde{l}-b,c) \right\} & \text{for } c>1. \end{cases} \tag{3.1}$$

Proof. For $h-i\le q$ it is obvious that no physical step has to be performed because $h-i$ recording steps and $h-i$ reverse steps suffice to reverse $h-i\le q$ physical steps.

For $h-i>q$ and $c=1$ one can perform only one checkpoint writing that copies states $i-q+1,\ldots,i$. To reverse $F_{h-q},\ldots,F_{h-1}$ the physical

steps $F_i, \ldots, F_{h-q-1}$ are evaluated. In oder to evaluate the next reverse steps $\bar{F}_{h-2q}, \ldots, \bar{F}_{h-q-1}$, one has to perform the physical steps $F_i, \ldots, F_{h-2q-1}$ and so on. Therefore it follows that

$$\begin{aligned} t(i,h,c) &= \\ &= t_0 + \cdots + t_{h-q-1} + t_0 + \cdots + t_{h-2q-1} + \cdots + t_0 + \cdots + t_{h-\lfloor (h-i)/q \rfloor q-1} \\ &\equiv \sum_{j=i}^{h-1} \Big\lfloor \frac{h-j-1}{q} \Big\rfloor t_j \ . \end{aligned}$$

If $h-i > q$ and $c > 1$ the first checkpoint writing copies states $i-q+1, \ldots, i$ and the second checkpoint writing copies states $\tilde{l}-q+1, \ldots, \tilde{l}$ with $i+q \leq \tilde{l} < h$. This second checkpoint must be kept until the $h-\tilde{l}$ physical steps on the right of state $\tilde{l}$ are reversed because of Lemma 3.1. These physical steps on the right-hand side of the second checkpoint have to be reversed with up to $c-1$ checkpoints at any time, taking the one storing states $\tilde{l}-q+1, \ldots, \tilde{l}$ into account. The cost of this reversal equals $t(\tilde{l}, h, c-1)$ because otherwise $t(i,h,c)$ would not be minimal. If $b > 0$ is valid during the release of the checkpoint storing states $\tilde{l}-q+1, \ldots, \tilde{l}$ the reverse steps $\bar{F}_{\tilde{l}-1}, \ldots, \bar{F}_{\tilde{l}-b}$ can be performed. The next physical step to be reversed is given by $\bar{F}_{\tilde{l}-b-1}$. Since it is assumed that q physical steps can be recorded on the tape the next forward sweep to reverse $F_{\tilde{l}-b-1}$ advance to state $\tilde{l}-b-1-q$ in order to minimize the number of physical steps performed and hence the cost of the reversal. One has $\tilde{l}-b-1-q < \tilde{l}-q+1$. Therefore the checkpoint storing the states $\tilde{l}-q+1, \ldots, \tilde{l}$ can be freed up because it cannot be applied to reduce the reversal cost.

The remaining $\tilde{l}-b$ physical steps are reversed with a maximum of c checkpoints at any time and the minimal cost $t(i, \tilde{l}-b, c)$ for the same reason as above. Furthermore, one has to perform a forward sweep that advances from state i to state $\tilde{l}$. Hence, it follows that $t(i,h,c)$ has the form

$$t(i,h,c) \quad = \quad \min_{i+q \leq \tilde{l} < h} \left\{ \sum_{j=i}^{\tilde{l}-1} t_j + t(\tilde{l}, h, c-1) + t(i, \tilde{l}-b, c) \right\} \ .$$

■

3.3 Multi-step Evolutions with Uniform Step Costs

Throughout this section it is supposed that the given q-step evolution F comprises the physical steps F_i, $0 \leq i < l$, with uniform step costs. Without loss of generality one may presume $t_i = 1$ for $0 \leq i < l$.

If $l = mq$, $m \in \mathbb{N}$, an obvious idea for reversing F could be to group always q states together to a *mega state*

$$\big((i-1)q+1, \ldots, iq\big) \quad \text{for} \quad i \in \{0, \ldots, m\} \ .$$

Subsequently one could define an one-step evolution $\mathbb{F}$ comprising the *mega steps* $\mathbb{F}_I$, $I = 0, \ldots, m$ acting on the mega states, such that the evaluation of

$\mathbb{F}_I$ is identical to the evaluation of the physical steps $F_i, \ldots, F_{i+q}$ of the original multi-step evolution F with $i = Iq$. Naturally, in order to reverse $\mathbb{F}$ it is possible to apply the known results for one-step evolutions with uniform step costs (see e.g. [GPRS96], [GW00]). Nevertheless, several degrees of freedom are neglected using this first approach, for example the possibility to perform a checkpoint writing that copies a mega state $\tilde{L} = (\tilde{l} - q + 1, \ldots, \tilde{l})$ such that $\tilde{l}/q \notin \mathbb{N}$. Furthermore, the number b of linear arguments is not taken into account. Therefore, the possibility to perform additional reverse steps during the release of a checkpoint is ignored. Hence it is not certain that the *mega stepping* yields an optimal serial reversal schedule. Therefore the next subsection studies the question how to construct serial reversal schedules that attain minimal cost in detail.

3.3.1 Derivation of Optimal Reversal Schedules

As mentioned already above only the time needed to perform physical steps may vary from one serial reversal schedule to another. In this section an explicit formula for the optimal temporal complexity of the reversal of F is derived. In addition to this minimization of the number of physical steps performed the number of checkpoint writings W_j is also minimized. Thus the run time of the reversal is reduced further if the temporal complexity of the action W_j is not really negligible.

In order to simplify the notation used in the proofs within this section the following function will be used extensively:

Definition 3.2 (Function $\beta(a, r)$). For given $a, r \in \mathbb{Z}$ define the function

$$\beta(a, r) \equiv \begin{cases} \binom{a+r}{a} & \text{if} \quad a \geq 0 \\ 0 & \text{else} \end{cases} .$$

It follows that for any fixed $a \in \mathbb{Z}$ the function $\beta(a, r)$ is monotonic in r. Furthermore $\beta(a, r)$ represents for any fixed $a \in \mathbb{Z}$ a polynomial in r. In order to prove the explicit formula for the minimal number of physical steps that have to be performed during the reversal it is advantageous to have the following assertion at hand.

Lemma 3.3.
For $b, c, q \in \mathbb{N}$ and $l \in \mathbb{R}$ let r be the unique integer such that the inequality

$$\begin{aligned} \gamma(c, r-1) \;<\; l \;\leq\; \gamma(c, r) \quad &\textit{with} \\ \gamma(c, r) \;\equiv\; \beta(c, r)q + (\beta(c-1, r) - 1)b & \end{aligned}$$

is fulfilled. Then the expression

$$rl - \beta(c+1, r-1)q - (\beta(c, r-1) - r)b$$

is for any $c > 0$ a piecewise linear and convex function of $l \geq 0$.

Proof. Define $f(l,c) \equiv rl - \beta(c+1,r-1)q - (\beta(c,r-1)-r)b$. The function f is piecewise linear since $\beta(c+1,r-1)q + (\beta(c,r-1)-r)b$ is constant for $l \in (\gamma(c,r-1),\gamma(c,r)]$. Hence f is continuous in l for all $l \neq \gamma(c,r)$, $r \in \mathbb{N}$. It will be shown that f is continuous for all $l \in \mathbb{R}$. Chose an arbitrary $r \in \mathbb{N}$. For $l_n \in \mathbb{R}$, $n \in \mathbb{N}$, with $l_n < \gamma(c,r)$ and $\lim_{n\to\infty} l_n = \gamma(c,r)$ if n approaches infinity one obtains for $f(c,l_n)$

$$\begin{aligned}\lim_{n\to\infty} f(c,l_n) &= \lim_{n\to\infty}(rl_n) - \beta(c+1,r-1)q - (\beta(c,r-1)-r)b \\ &= r\gamma(c,r) - \beta(c+1,r-1)q - (\beta(c,r-1)-r)b\,.\end{aligned}$$

For $l_n \in \mathbb{R}$, $n \in \mathbb{N}$, with $l_n > \gamma(c,r)$ and $\lim_{n\to\infty} l_n = \gamma(c,r)$ it follows for the limit of $f(c,l_n)$ that

$$\begin{aligned}\lim_{n\to\infty} f(c,l_n) &= \lim_{n\to\infty}\big((r+1)l_n\big) - \beta(c+1,r)q - (\beta(c,r)-r-1)b \\ &= (r+1)\gamma(c,r) - \beta(c+1,r)q - (\beta(c,r)-r-1)b \\ &= r\gamma(c,r) - \beta(c+1,r-1)q - (\beta(c,r-1)-r)b\,.\end{aligned}$$

Therefore f is continuous in $l = \gamma(c,r)$. Since an arbitrary $r \in \mathbb{N}$ was considered one obtains that f is continuous for all $l \in \mathbb{R}$.

The function $\gamma(c,r)$ is strictly monotone increasing in r. I.e. for $l_1, l_2 \in \mathbb{R}$ with $l_1 \leq l_2$ there exist $r_1, r_2 \in \mathbb{N}$ with

$$\gamma(c,r_1-1) \;<\; l_1 \;\leq\; \gamma(c,r_1), \quad \gamma(c,r_2-1) \;<\; l_2 \;\leq\; \gamma(c,r_2),$$

and $r_1 \leq r_2$. Hence, on can conclude that the slope of f is monotone increasing. Therefore, f is a continuous, piecewise linear function the slope of which increases monotonously. This yields that f is convex in l, which proves the assertion. ∎

If one needs to reverse the evolutionary system F a very natural aim is to execute the reversal at the lowest cost that can be achieved. The number of recording steps and reverse steps is determined already by the number of physical steps l to be reversed as mentioned before. Two variables that can be minimized are given by the number of physical steps F_i performed during the reversal and the number of checkpoint writings performed during the reversal. Now the former will be considered in detail. Because of the assumption that $t_i = 1$ for all physical steps F_i, $0 \leq i < l$, the function $t(i,h,c)$ determining the minimal cost to reverse the physical steps between state i and state h using no more than c checkpoints can be denoted by $t(h-i,c)$. In the case of uniform step costs only the number of physical steps to be reversed has an influence on the cost $t(i,h,c)$, but not the particular physical steps to be reversed. Furthermore, $t(h-i,c)$ equals the number of physical steps F_i performed during the reversal since $t_i = 1$ for all $0 \leq i < l$ is assumed. The following theorem determines an explicit formula for $t(l,c)$. In the proof serial reversal schedules are constructed which attain the minimal cost $t(l,c)$.

Theorem 3.1 (Minimal Number of Performed Physical Steps).
Suppose the q-step evolution F under consideration comprises l physical steps with uniform step costs and determines the number b of linear arguments according to Def. 2.1. Then the function

$$t(l,c) \;\equiv\; \begin{cases} 0 & \text{for} \quad l \leq q, \\ (\lceil l/q \rceil - 1)\big(l - q\lceil l/q \rceil/2\big) & \text{for} \quad c = 1, l > q, \\ \min\limits_{q \leq \tilde{l} < h} \{\tilde{l} + t(l - \tilde{l}, c-1) + t(\tilde{l} - b, c)\} & \text{for} \quad c > 1, l > q, \end{cases} \tag{3.2}$$

denoting the minimal cost to reverse a sequence of l physical steps with uniform step costs storing up to c checkpoints at any time takes the explicit form

$$t(l,c) \;=\; rl - \beta(c+1, r-1)q - (\beta(c, r-1) - r)b, \tag{3.3}$$

where r is the unique integer satisfying

$$\beta(c, r-1)q + (\beta(c-1, r-1) - 1)b < l \leq \beta(c,r)q + (\beta(c-1,r) - 1)b\,. \tag{3.4}$$

Proof. Because of Lemma 3.2 one can conclude that the function $t(l,c)$ satisfies (3.2). Now Identity (3.3) is proven by induction on c and l:

$c = 1$ and $l \in \mathbb{N}$:
With the identity

$$\gamma(c,r) \;\equiv\; \beta(c,r)q + (\beta(c-1,r) - 1)b$$

the inequalities

$$\gamma(1, r-1) \;=\; rq \;<\; l \;\leq\; (r+1)q \;=\; \gamma(1,r)$$

hold, if and only if $r \equiv \lceil l/q \rceil - 1$. This yields

$$\begin{aligned} t(l,1) &= (\lceil l/q \rceil - 1)\big(l - q\lceil l/q \rceil/2\big) \\ &= (\lceil l/q \rceil - 1)l - \beta(2, \lceil l/q \rceil - 2)q - (\lceil l/q \rceil - 1 - \lceil l/q \rceil + 1)b \\ &= rl - \beta(2, r-1)q - (\beta(1, r-1) - r)b\,. \end{aligned}$$

Hence, Identity (3.3) has been proven.

$c \in \mathbb{N}$ and $l \leq q$:
One finds that $r = 0$ is the only integer satisfying Inequality (3.4) because

$$\gamma(c, -1) \;=\; -b \;<\; l \;\leq\; q = \;\gamma(c,0)\,.$$

Therefore, it follows that

$$t(l,c) \;=\; 0 \;=\; 0 \cdot l - \beta(c+1, -1)q - (\beta(c,-1) - 0)b$$

as asserted.

Induction step in lexicographical order of (c,l):
The numbers $c > 1$ and $l > q$ are given. It is clear that they determine a unique

$r \in \mathbb{N}$ satisfying Inequality (3.4). Suppose that the assertion is true for all $(\tilde{c}, \tilde{l}\,)$ with $\tilde{c} < c$ or $\tilde{c} = c$ and $\tilde{l} < l$.

One has to decide which states $\tilde{l}-q+1, \ldots, \tilde{l}$ are copied to the second checkpoint. It will be shown that Equation (3.3) is fulfilled if the second checkpoint writing copies states satisfying certain conditions. Then it will be proven that more than $rl - \beta(c+1, r-1)q - (\beta(c, r-1) - r)b$ physical steps are performed if the second checkpoint stores states that do not satisfy these conditions.

Let the second checkpoint writing copy states $\hat{l} - q + 1, \ldots, \hat{l}$ such that the inequalities

$$\gamma(c, r-2) \;\leq\; \hat{l} - b \;\leq\; \gamma(c, r-1) \qquad \text{and} \tag{3.5}$$

$$\gamma(c-1, r-1) \;\leq\; l - \hat{l} \;\leq\; \gamma(c-1, r) \tag{3.6}$$

are valid. These conditions are fulfilled if and only if

$$\max\{\gamma(c, r-2) + b, l - \gamma(c-1, r)\} \leq \hat{l} \leq \min\{\gamma(c, r-1) + b, l - \gamma(c-1, r-1)\}.$$

It is always possible to find $\hat{l}$ satisfying Inequalities (3.5) and (3.6) because

1. $\gamma(c, r-2) + b \;\leq\; \gamma(c, r-1) + b$:

   $0 \leq \beta(c-1, r-1)q + \beta(c-2, r-1)b \;\Rightarrow$
   $0 \leq (\beta(c, r-1) - \beta(c, r-2))q + (\beta(c-1, r-1) - \beta(c-1, r-2))b \;\Rightarrow$
   $0 \leq \gamma(c, r-1) - \gamma(c, r-2)$,

2. $\gamma(c, r-2) + b \;\leq\; l - \gamma(c-1, r-1)$:

   $\gamma(c, r-1) \leq l \;\Rightarrow$
   $(\beta(c, r-2) + \beta(c-1, r-1))q + (\beta(c-1, r-2) + \beta(c-2, r-1) - 1)b \leq l \Rightarrow$
   $\gamma(c, r-2) + b \leq l - \gamma(c-1, r-1)$,

3. $l - \gamma(c-1, r) \;\leq\; \gamma(c, r-1) + b$:

   $l \leq \gamma(c, r) \;\Rightarrow$
   $l \leq (\beta(c, r-1) + \beta(c-1, r))q + (\beta(c-1, r-1) + \beta(c-2, r) - 1)b \;\Rightarrow$
   $l - \gamma(c-1, r) \;\leq\; \gamma(c, r-1) + b$,

4. $l - \gamma(c-1, r) \;\leq\; l - \gamma(c-1, r-1)$:

   $0 \leq \beta(c-2, r)q + \beta(c-3, r)b \;\Rightarrow$
   $0 \leq (\beta(c-1, r) - \beta(c-1, r-1))q + (\beta(c-2, r) - \beta(c-2, r-1))b \;\Rightarrow$
   $0 \leq \gamma(c-1, r) - \gamma(c-1, r-1)$.

It follows from Inequality (3.6) and the induction hypothesis that the minimal number of physical steps performed for reversing $l - \hat{l}$ of them is given by $t(l - \hat{l}, c-1)$. During the release of the second checkpoint b reverse steps can be performed. From Inequality (3.5) and the induction hypothesis one obtains that

the number of physical steps performed is no less than $t(\hat{l}-b,c)$ for reversing $\hat{l}-b$ physical steps. Furthermore, one needs to perform $\hat{l}$ physical steps to go to state $\hat{l}$. Hence, if the second checkpoint writing copies states $\hat{l}-q+1,\ldots,\hat{l}$ with $\hat{l}$ satisfying Inequalities (3.5) and (3.6), the minimal number of physical steps performed while reversing l physical steps equals

$$\hat{l}+t(l-\hat{l},c-1)+t(\hat{l}-b,c)=rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b\;. \quad (3.7)$$

It remains to prove that the number of physical steps performed for the reversal of l physical steps becomes greater than (3.7) if the second checkpoint stores states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}$ not satisfying Inequalities (3.5) and (3.6).

The expression $rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b$ is convex in l because of Lemma 3.3. Therefore,

$$\tilde{l}+t(l-\tilde{l},c-1)+t(\tilde{l}-b,c)$$

is also convex in $\tilde{l}$ with $\tilde{l}\in\{q,\ldots,l-1\}$. A convex function can have at most one interval where the function value is minimal. For that reason it is sufficient to prove that the number of physical steps performed to reverse l physical steps becomes greater than $rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b$ if the second checkpoint writing copies states $\tilde{l}-q+1,\ldots,\tilde{l}$ with

$$\begin{aligned}\tilde{l}&=\max\{\gamma(c,r-2)+b,l-\gamma(c-1,r)\}-1 \qquad\qquad \text{or}\\ \tilde{l}&=\min\{\gamma(c,r-1)+b,l-\gamma(c-1,r-1)\}+1\;.\end{aligned}$$

First, the case $\tilde{l}=\max\{\gamma(c,r-2)+b,l-\gamma(c-1,r)\}-1$ will be examined. Let $\gamma(c,r-2)+b$ be smaller than $l-\gamma(c-1,r)$. It follows that

$$\begin{gathered}\gamma(c,r-2)\;\le\;\tilde{l}-b\;=\;l-\gamma(c-1,r)-1-b\;<\;\gamma(c,r-1)\quad\text{and}\\ \gamma(c-1,r)\;<\;l-\tilde{l}\;\le\;\gamma(c-1,r+1)\;.\end{gathered}$$

Using the induction hypothesis one obtains that at least $t(\tilde{l}-b,c)$ physical steps have to be performed to reverse the left-hand side of $\tilde{l}-b$ and at least $t(l-\tilde{l},c-1)$ physical steps have to be performed to reverse the right-hand side of $\tilde{l}$. One has

$$\begin{aligned}&rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b-\tilde{l}-t(l-\tilde{l},c-1)-t(\tilde{l}-b,c)\;=\\ &\quad=\;rl-\tilde{l}-(r-1)(\tilde{l}-b)-(r+1)(l-\tilde{l})-(\beta(c+1,r-1)-\beta(c+1,r-2)\\ &\qquad-\beta(c,r))q-(\beta(c,r-1)-r-\beta(c,r-2)+r-1-\beta(c-1,r)+r+1)b\\ &\quad=\;-\gamma(c-1,r)-1+\beta(c-1,r)q+(\beta(c-2,r)-1)b\;=\;-1\;.\end{aligned}$$

Therefore, more than (3.7) physical steps have to be performed to reverse l physical steps if the states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}=l-\gamma(c-1,r)-1$ are chosen as second checkpoint. Now assume that

$$\gamma(c,r-2)+b\;\ge\;l-\gamma(c-1,r)\;.$$

Then the inequalities

$$\begin{gathered}\gamma(c,r-3)\;\le\;\tilde{l}-b\;=\;\gamma(c,r-2)-1\;<\;\gamma(c,r-2)\quad\text{and}\\ \gamma(c-1,r-1)\;<\;l-\tilde{l}\;\le\;\gamma(c-1,r)+1\end{gathered}$$

are fulfilled. Hence, the number of physical steps performed to reverse the right-hand side of $\tilde{l}$ is not smaller than $r(l-\tilde{l})-\beta(c,r-1)q-(\beta(c-1,r-1)-r)b$ and the number of physical steps performed to reverse the left-hand side of $\tilde{l}-b$ equals $t(\tilde{l}-b,c)$. Therefore, one has

$$\begin{aligned} & rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b-\tilde{l}-t(l-\tilde{l},c-1)-t(\tilde{l}-b,c) \;\leq \\ & \quad\leq\; rl-\tilde{l}-(r-2)(\tilde{l}-b)-r(l-\tilde{l})-(\beta(c+1,r-1)-\beta(c+1,r-3) \\ & \qquad -\beta(c,r-1))q-(\beta(c,r-1)-r-\beta(c,r-3)+r-2-\beta(c-1,r-1)+r)b \\ & \quad=\; \gamma(c,r-2)+b-1-\beta(c,r-2)q-\beta(c-1,r-2)b \;=\; -1\,. \end{aligned}$$

Consequently, more than (3.7) physical steps are performed to reverse l of them if the second checkpoint stores states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}=\gamma(c,r-2)+b-1$.

It is shown that for the reversal of l physical steps using states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}=\max\{\gamma(c,r-2)+b,l-\gamma(c-1,r)\}-1$ as second checkpoint leads to more than (3.7) physical steps performed during the reversal process. It is left to consider the case $\tilde{l}=\min\{\gamma(c,r-1)+b,l-\gamma(c-1,r-1)\}+1$.

One possibility is that $\gamma(c,r-1)+b<l-\gamma(c-1,r-1)$. Then one obtains

$$\begin{gathered} \gamma(c,r-1) \;<\; \tilde{l}-b \;=\; \gamma(c,r-1)+1 \;\leq\; \gamma(c,r) \quad \text{and} \\ \gamma(c-1,r-1) \;\leq\; l-\tilde{l} \;\leq\; \gamma(c-1,r)\,. \end{gathered}$$

Therefore, $t(\tilde{l}-b,c)$ physical steps are performed to reverse the left-hand side of $\tilde{l}-b$ and $t(l-\tilde{l},c-1)$ physical steps are performed to reverse the right-hand side of $\tilde{l}$. It yields

$$\begin{aligned} & rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b-\tilde{l}-t(l-\tilde{l},c-1)-t(\tilde{l}-b,c) \;= \\ & \quad=\; rl-\tilde{l}-r(\tilde{l}-b)-r(l-\tilde{l})-(\beta(c+1,r-1)-\beta(c+1,r-1) \\ & \qquad -\beta(c,r-1))q-(\beta(c,r-1)-r-\beta(c,r-1)+r-\beta(c-1,r-1)+r)b \\ & \quad=\; -\tilde{l}+\beta(c,r-1)q+\beta(c-1,r-1)b \;=\; -1\,. \end{aligned}$$

It follows that more than (3.7) physical steps performed are required to reverse l physical steps if the second checkpoint writing copies states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}=\gamma(c,r-1)+b+1$. Now consider the second possibility, namely $\gamma(c,r-1)+b\geq l-\gamma(c-1,r-1)$. One obtains

$$\begin{gathered} \gamma(c,r-2) \;<\; \tilde{l}-b \;=\; l-\gamma(c-1,r-1)+1-b \;\leq\; \gamma(c,r) \quad \text{and} \\ \gamma(c-1,r-2) \;\leq\; l-\tilde{l} \;<\; \gamma(c-1,r-1)\,. \end{gathered}$$

Building on the induction hypothesis the number of physical steps performed for the reversal of the right-hand side of $\tilde{l}$ equals $t(l-\tilde{l},c-1)$ and the number of physical steps performed for the reversal of the left-hand side of $\tilde{l}-b$ is not smaller than $(r-1)(\tilde{l}-b)-\beta(c+1,r-2)q-(\beta(c,r-2)-r+1)b$. Moreover

$$\begin{aligned} & rl-\beta(c+1,r-1)q-(\beta(c,r-1)-r)b-\tilde{l}-t(l-\tilde{l},c-1)-t(\tilde{l}-b,c)\leq \\ & \quad\leq\; rl-\tilde{l}-(r-1)(\tilde{l}-b)-(r-1)(l-\tilde{l})-(\beta(c+1,r-1)-\beta(c+1,r-2) \\ & \qquad -\beta(c,r-2))q-(\beta(c,r-1)-\beta(c,r-2)-\beta(c-1,r-2)+r-2)b \\ & \quad=\; l-\tilde{l}-\beta(c-1,r-1)q-(\beta(c-2,r-1)-1)b \;=\; -1\,. \end{aligned}$$

For that reason more than (3.7) physical steps are performed to reverse l physical steps if states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}=l-\gamma(c-1,r-1)+1$ are chosen as second checkpoint.

Finally, Equation (3.3) is proven because the number of physical steps performed to reverse l of them exceeds (3.7) if the second checkpoint writing copies states $\tilde{l}-q+1,\ldots,\tilde{l}$ with $\tilde{l}$ not satisfying the Inequalities (3.5) and (3.6). ∎

Figure 3.8 illustrates the behavior of the function $t(l,c)$ for different values of c if the parameters q and b of the multi-step evolution under consideration have the values 2 and 1, respectively. Obviously, this function is piecewise linear and strictly monotonous increasing in l for each value of c. Hence for a

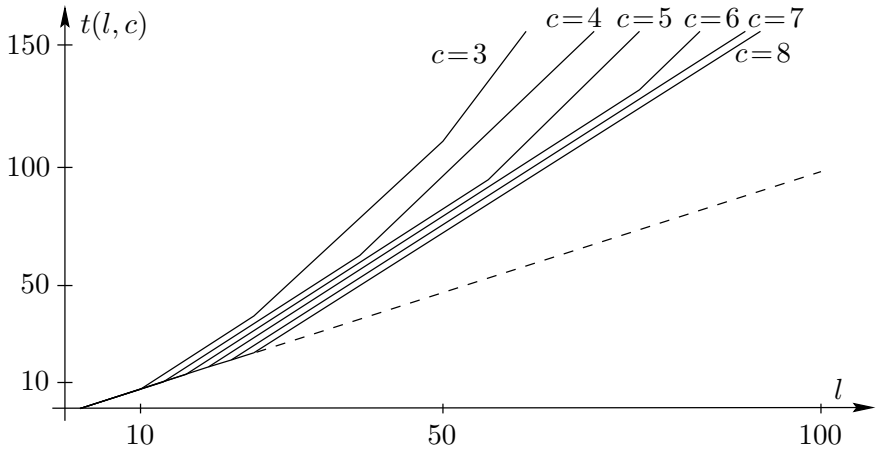


Figure 3.8: Values of $t(l,c)$ for $q=2$ and $b=1$

given increase in run time that is acceptable one finds easily the corresponding minimal number of checkpoints needed. Furthermore, for a fixed number of checkpoints that are available the minimal increase in run time that is not avoidable can be determined.

Table 3.1 serves to illustrate the gain that can be achieved by taking $b>0$ into account. To that end the value of $t(l,c)$ is calculated for $q=2$, $b=1$,

	$q=2$		$q=4$	
l	$c=4$	$c=6$	$c=4$	$c=6$
100	10.8 %	11.0 %	16.7 %	8.9 %
250	6.5 %	10.5 %	12.0 %	12.4 %
500	7.8 %	8.7 %	9.8 %	15.7 %
1000	5.2 %	6.4 %	10.1 %	10.9 %
1500	5.1 %	6.1 %	8.9 %	10.0 %
2000	4.4 %	5.8 %	7.7 %	8.7 %
2500	4.6 %	6.5 %	8.4 %	10.7 %
3000	4.6 %	5.2 %	7.3 %	9.2 %

Table 3.1: Achievable Improvements Taking $b>0$ Into Account

$c \in \{4, 6\}$ and for $q = 4$, $b = 3$, $c \in \{4, 6\}$. The reduction of the cost $t(l, c)$ is reported as percentage of the cost $t(l, c)$ if one sets $b = 0$ neglecting the additional reverse steps that can be performed during the release of a checkpoint. Assume $l = 1000$, $q = 4$, $b = 3$, and $c = 6$. If one takes the possibility to perform three additional reverse steps during the release of each checkpoint into account the number of physical steps performed during the reversal is reduced by 10.9 %.

The structure of the valid domain for the value $\hat{l}$ is displayed by Fig. 3.9. Usually, i.e. if c and r have nontrivial values, there is a wide range of choices. This fact can be used for a further improvement of the serial reversal schedules.

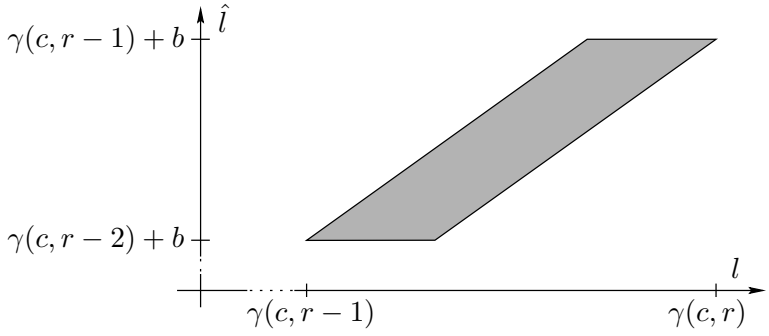


Figure 3.9: Domain of $\hat{l}$ Optimizing the Number of Physical Steps

As mentioned above the number of times current states are copied to a checkpoint during the reversal may vary from one serial reversal schedule to the other. Hence one could look for serial reversal schedules, which minimize the total number of checkpoint writings during the reversal under the precondition that the number of physical steps performed by the serial reversal schedules under consideration is optimal, i.e. minimal. These optimal serial reversal schedules are constructed in the proof of the following theorem.

Theorem 3.2 (Minimal Number of Checkpoint Writings).
Suppose, the given q-step evolution F comprises l physical steps and determines the corresponding value of b. Among all serial reversal schedules requiring $t(l, c)$ physical steps to be performed for the reversal of the given l physical steps with up to c checkpoints at any time the minimal number of checkpoint writings is given for all $l \in \mathbb{N}$ by

$$s(l,c) \equiv \begin{cases} 0 & \text{if } l \le q \\ \beta(c-1, r-1) & \text{if } q < l \le \phi(c,r)q + (\beta(c-1,r-1)-1)b \\ m_1 - \beta(c, r-1) & \text{if } q < (m_1-1)q + (m_2-1)b < l \le m_1 q + m_2 b \\ & \text{with } \phi(c,r) + 1 \le m_1 \le \beta(c,r) \text{ and} \\ & m_2 = m_1 - \beta(c, r-1) - 1 \,, \end{cases}$$

where r is again the unique integer satisfying

$$\beta(c, r-1)q + (\beta(c-1, r-1) - 1)b < l \le \beta(c, r)q + (\beta(c-1, r) - 1)b \quad (3.8)$$

and

$$\phi(c, r) \equiv \beta(c, r-1) + \beta(c-1, r-1) \,.$$

Proof. If $l \leq q$ no checkpoint writing is performed because only $l \leq q$ recording steps and $l \leq q$ reverse steps yield the desired reversal. To prove the assertion for $l > q$ once again an induction on c and l is used:

$c = 1$ and $l \in \mathbb{N}$, $l > q$:
The states $1 - q, \ldots, 0$ are copied to the only checkpoint once. Therefore the number of checkpoint writings equals 1. Furthermore, using the definitions

$$\begin{aligned} \gamma(c,r) &\equiv \beta(c,r)q + (\beta(c-1,r)-1)b \qquad \text{and} \\ \eta(c,r) &\equiv \phi(c,r+1)q + (\beta(c-1,r)-1)b \end{aligned}$$

one has

$$\gamma(1,r-1) \;=\; rq \;<\; l \;\leq\; (r+1)q \;=\; \eta(1,r-1) \;=\; \gamma(1,r)$$

if and only if $r \equiv \lceil l/q \rceil - 1$. It follows that $s(l,1) = \beta(0,r-1) = 1$.

$c \geq 1$ and $q < l \leq 2q$:
The states $1 - q, \ldots, 0$ are copied to the only checkpoint once. Now $l - q$ physical steps are performed. After that q recording steps and q reverse steps are evaluated. Then the states stored in the only checkpoint are read. Subsequently $l - q$ recording steps and $l - q$ reverse steps yield the desired reversal. Hence, the number of checkpoint writings equals 1. Furthermore, the inequality

$$\gamma(c,0) \;=\; q \;<\; l \;\leq\; 2q \;\leq\; \eta(c,0)$$

holds. Thus $s(l,c) = \beta(c-1,0) = 1$.

Induction step in lexicographical order of (c,l):
The numbers $c > 1$ and $l > 2q$ are given. It is clear that they determine a unique $r \in \mathbb{N}$ satisfying Inequality (3.4). Suppose, the assertion is true for all $(\tilde{c}, \tilde{l}\,)$ with $\tilde{c} < c$ or $\tilde{c} = c$ and $\tilde{l} < l$. First, the assertion will be proven for $l \leq \eta(c,r-1)$. The second possibility, namely $l > \eta(c,r-1)$, will be examined later. In both cases it will be shown that the number of checkpoint writings equals $s(l,c)$ if the states that are copied to the second checkpoint satisfy certain conditions. In a second step it will be proven that there is no possibility to achieve a smaller number of checkpoint writings.

If $l \leq \eta(c,r-1)$ then copy states $\hat{l} - q + 1, \ldots, \hat{l}$ to the second checkpoint such that

$$\gamma(c,r-2) \;\leq\; \hat{l} - b \;\leq\; \eta(c,r-2) \qquad \text{and} \tag{3.9}$$

$$\gamma(c-1,r-1) \;\leq\; l - \hat{l} \;\leq\; \eta(c-1,r-1)\,. \tag{3.10}$$

It is always possible to find an appropriate $\hat{l}$, because

1. $\gamma(c,r-2) + b \;\leq\; \eta(c,r-2) + b$:

   $0 \leq \beta(c-1,r-2)q \;\Rightarrow\; 0 \leq (\phi(c,r-1) - \beta(c,r-2))q \;\Rightarrow$
   $0 \leq \eta(c,r-2) - \gamma(c,r-2)$,

2. $\gamma(c, r-2) + b \;\leq\; l - \gamma(c-1, r-1)$:

   $\gamma(c, r-1) \leq l \quad \Rightarrow \quad \gamma(c, r-2) + \gamma(c-1, r-1) + b \leq l$,

3. $l - \eta(c-1, r-1) \;\leq\; \eta(c, r-2) + b$:

   $l \leq \eta(c, r-1) \;\Rightarrow$
   $l \leq (\phi(c, r-1) + \phi(c-1, r))q + (\beta(c-2, r-1) + \beta(c-1, r-2) - 1)b \Rightarrow$
   $l - \eta(c-1, r-1) \leq \; \eta(c, r-2) + b$,

4. $l - \eta(c-1, r-1) \;\leq\; l - \gamma(c-1, r-1)$:

   $0 \leq \beta(c-2, r-1)q \;\Rightarrow \quad 0 \leq (\phi(c-1, r) - \beta(c-1, r-1))q \;\Rightarrow$
   $0 \leq \eta(c-1, r-1) - \gamma(c-1, r-1)$.

From the induction hypothesis and Inequality (3.9) follows that no less than $\beta(c-1, r-2)$ checkpoint writings are needed for reversing the left-hand side of state $\hat{l} - b$. The induction hypothesis and Inequality (3.10) yield that the minimal number of checkpoint writings equals $\beta(c-2, r-1)$ for reversing the right-hand side of $\hat{l}$. Hence, the minimal number of checkpoint writings is equals

$$\beta(c-1, r-2) + \beta(c-2, r-1) \;=\; \beta(c-1, r-1)$$

for reversing l physical steps if states $\hat{l} - q + 1, \ldots, \hat{l}$ with $\hat{l}$ satisfying Inequalities (3.9) and (3.10) are copied to the second checkpoint.

It will be shown that more than $\beta(c-1, r-1)$ checkpoint writings are needed if states $\tilde{l} - q + 1, \ldots, \tilde{l}$ with $\tilde{l}$ not satisfying Inequalities (3.9) and (3.10) are copied to the second checkpoint. There are four possibilities.

First, let $\tilde{l}$ satisfy Inequality (3.9) but $\eta(c-1, r-1) < l - \tilde{l}$. Then at least $\beta(c-1, r-2)$ checkpoint writings are performed for reversing the left-hand side of $\tilde{l} - b$ and at least $s(l - \tilde{l}, c-1) > \beta(c-2, r-1)$ checkpoint writings are needed for reversing the right-hand side of $\tilde{l}$. Hence, their minimal number equals

$$\beta(c-1, r-2) + s(l - \tilde{l}, c-1) > \beta(c-1, r-1) \; .$$

Second, let $\tilde{l}$ satisfy Inequality (3.10) while $\eta(c, r-2) < \tilde{l} - b$. It follows from the induction hypothesis that the number of checkpoint writings is no less than $s(\tilde{l} - b, c) > \beta(c-1, r-2)$ to reverse the left-hand side of $\tilde{l} - b$ and at least $\beta(c-2, r-1)$ checkpoint writings are needed to reverse the right-hand side of $\tilde{l}$. Therefore, at least

$$s(\tilde{l} - b, c) + \beta(c-2, r-1) > \beta(c-1, r-1)$$

checkpoint writings are performed. Third, if

$$\tilde{l} - b \;\; > \;\; \eta(c, r-2) \quad \text{and} \quad l - \tilde{l} \; > \; \eta(c-1, r-1)$$

it follows that

$$l \;=\; l - \tilde{l} + \tilde{l} \; > \; \eta(c-1, r-1) + \eta(c, r-2) + b \; > \; \eta(c, r-1) \; .$$

This is a contradiction to the assumption that $l \leq \eta(c, r-1)$. Finally, if

$$\gamma(c, r-2) \;>\; \tilde{l} - b \quad \text{and/or} \quad \gamma(c-1, r-1) \;>\; l - \tilde{l}$$

the number of physical steps performed is not optimal because of Theorem 3.1. This is a contradiction to the assumption that the serial reversal schedule under consideration needs the minimal number $t(l, c)$ of physical steps performed to reverse the l given physical steps. Therefore it is shown that the minimal number of checkpoint writings equals $s(l, c) = \beta(c-1, r-1)$ if $l \leq \eta(c, r-1)$.

Now the assertion will be proven for $l > \eta(c, r-1)$. There exists exactly one $m_1 \in \{\phi(c, r)+1, \ldots, \beta(c, r)\}$ with $(m_1-1)q + (m_2-1)b < l \leq m_1 q + m_2 b$, where $m_2 \equiv m_1 - \beta(c, r-1) - 1$. Choose $\hat{m}_1 \in \{\phi(c, r-1)+1, \ldots, \beta(c, r-1)\}$ such that $m_1 - \hat{m}_1 \in \{\phi(c-1, r), \ldots, \beta(c-1, r)\}$. This is always possible. For example it is easy to check that

$$\hat{m}_1 \;\equiv\; \begin{cases} m_1 - \beta(c-1, r) & \text{if } \phi(c, r) + \beta(c-3, r) < m_1 \leq \beta(c, r) \\ \phi(c, r-1) + 1 & \text{else} \end{cases} \tag{3.11}$$

has the desired property. Copy states $\hat{l} - q + 1, \ldots, \hat{l}$ to the second checkpoint such that $\hat{l}$ satisfies the inequalities

$$(\hat{m}_1 - 1)q + (\hat{m}_2 - 1)b < \hat{l} - b \leq \hat{m}_1 q + \hat{m}_2 b \qquad \text{and} \tag{3.12}$$

$$(m_1 - \hat{m}_1 - 1)q + (m_2 - \hat{m}_2 - 2)b < l - \hat{l} \;\leq (m_1 - \hat{m}_1)q + (m_2 - \hat{m}_2 - 1)b \tag{3.13}$$

with $\hat{m}_2 \equiv \hat{m}_1 - \beta(c, r-2) - 1$. It is always possible to find an appropriate $\hat{l}$ because

1. $(\hat{m}_1 - 1)q + \hat{m}_2 b \;<\; \hat{m}_1 q + (\hat{m}_2 + 1)b$: Is obviously valid,
2. $(\hat{m}_1 - 1)q + \hat{m}_2 b \;<\; l - (m_1 - \hat{m}_1 - 1)q - (m_2 - \hat{m}_2 - 2)b - 1$:

   $$\begin{aligned} l \;&>\; (m_1 - 1)q + (m_2 - 1)b \;>\; (m_1 - 2)q + (m_2 - 2)b + 1 \;\Rightarrow \\ 0 \;&<\; l - (m_1 - \hat{m}_1 - 1)q - (m_2 - \hat{m}_2 - 2)b - 1 - (\hat{m}_1 - 1)q - \hat{m}_2 b\,, \end{aligned}$$

3. $l - (m_1 - \hat{m}_1)q - (m_2 - \hat{m}_2 - 1)b \;<\; l - (m_1 - \hat{m}_1 - 1)q - (m_2 - \hat{m}_2 - 2)b$: Is obviously valid,
4. $l - (m_1 - \hat{m}_1)q - (m_2 - \hat{m}_2 - 1)b \;\leq\; \hat{m}_1 q + (\hat{m}_2 + 1)b$:

   $$\begin{aligned} l \;&\leq\; m_1 q + m_2 b \;\Rightarrow \\ l \;&\leq\; (m_1 - \hat{m}_1)q + (m_2 - \hat{m}_2 - 1)b + \hat{m}_1 q + (\hat{m}_2 + 1)b\,. \end{aligned}$$

Furthermore, with $\tilde{m}_1 \equiv m_1 - \hat{m}_1$ and

$$\begin{aligned} \tilde{m}_2 \;&\equiv\; m_2 - \hat{m}_2 - 1 \;=\; m_1 - \hat{m}_1 - \beta(c, r-1) - 1 + \beta(c, r-2) + 1 - 1 \\ &=\; m_1 - \hat{m}_1 - \beta(c-1, r-1) - 1 \end{aligned}$$

Inequality (3.13) is equivalent to

$$(\tilde{m}_1 - 1)q + (\tilde{m}_2 - 1)b \;<\; l - \hat{l} \;\leq\; \tilde{m}_1 q + \tilde{m}_2 b\,.$$

Hence, one obtains from the induction hypothesis that $m_1 - \hat{m}_1 - \beta(c-1, r-1)$ checkpoint writings are needed to reverse the right-hand side of $\hat{l}$. Furthermore, it follows that the minimal number of checkpoint writings needed for reversing the left-hand side of $\hat{l} - b$ equals $\hat{m}_1 - \beta(c, r-2)$. Therefore, at least

$$\hat{m}_1 - \beta(c, r-2) + m_1 - \hat{m}_1 - \beta(c-1, r-1) \quad = \quad m_1 - \beta(c, r-1)$$

checkpoint writings are performed if $\hat{l} - q + 1, \ldots, \hat{l}$ are copied to the second checkpoint.

It is left to show that no less than $m_1 - \beta(c, r-1)$ checkpoint writings are needed if states $\tilde{l} - q + 1, \ldots, \tilde{l}$ with $\tilde{l}$ not satisfying Inequalities (3.12) and (3.13) are copied to the second checkpoint. There are eight possibilities to check.

1. Consider the case where Inequality (3.12) is fulfilled for $\tilde{l}$ but

$$l - \tilde{l} \; > \; (m_1 - \hat{m}_1)q + (m_2 - \hat{m}_2 - 1)b \; .$$

Then $s(l - \hat{l}, c-1)$ is greater than $m_1 - \hat{m}_1 - \beta(c-1, r-1)$ and the minimal number of checkpoint writings to reverse the given l physical steps equals

$$\hat{m}_1 - \beta(c, r-2) + s(l - \tilde{l}, c-1) \quad > \quad m_1 - \beta(c, r-1) \; .$$

2. Let Inequality (3.12) be valid for $\tilde{l}$ but $l - \tilde{l} \leq (\tilde{m}_1 - 1)q + (\tilde{m}_2 - 1)b$. It follows that

$$\begin{aligned} l \;&=\; \tilde{l} + l - \tilde{l} \;\leq\; \hat{m}_1 q + (\hat{m}_2 + 1)b + (m_1 - \hat{m}_1 - 1)q + (m_2 - \hat{m}_2 - 2)b \\ &=\; (m_1 - 1)q + (m_2 - 1)b \; . \end{aligned}$$

This is a contradiction to the assumption $(m_1 - 1)q + (m_2 - 1)b < l$.

3. Assume $\tilde{l}$ satisfies Inequality (3.13) while $\tilde{l} > \hat{m}_1 q + (\hat{m}_2 + 1)b$. For reversing the left-hand side of $\tilde{l} - b$ more than $\hat{m}_1 - \beta(c, r-2)$ checkpoint writings are needed. Hence, the minimal number of checkpoint writings to reverse the given l physical steps equals

$$s(\tilde{l} - b, c) + m_1 - \hat{m}_1 - \beta(c-1, r-1) \quad > \quad m_1 - \beta(c, r-1) \; .$$

4. Let $\tilde{l}$ satisfy Inequality (3.13) and $\tilde{l} \leq (\hat{m}_1 - 1)q + \hat{m}_2 b$. Then

$$\begin{aligned} l \;&\leq\; (m_1 - \hat{m}_1)q + (m_2 - \hat{m}_2 - 1)b + \tilde{l} \\ &\leq\; (m_1 - \hat{m}_1)q + (m_2 - \hat{m}_2 - 1)b + (\hat{m}_1 - 1)q + \hat{m}_2 b \\ &=\; (m_1 - 1)q + (m_2 - 1)b \; . \end{aligned}$$

This is a contradiction to the assumption $(m_1 - 1)q + (m_2 - 1)b < l$.

5. For $\tilde{l} \leq (\hat{m}_1 - 1)q + \hat{m}_2\, b$ and $l - \tilde{l} \leq (\tilde{m}_1 - 1)q + (\tilde{m}_2 - 1)b$ follows

$$\begin{aligned} l \;&=\; \tilde{l} + l - \tilde{l} \;\leq\; (\hat{m}_1 - 1)q + \hat{m}_2 b + (\tilde{m}_1 - 1)q + (\tilde{m}_2 - 1)b \\ &=\; (m_1 - 2)q + (m_2 - 2)b \; , \end{aligned}$$

which yields the same contradiction as in 4.

6. Studying the case $\tilde{l} > \hat{m}_1 q + (\hat{m}_2 + 1)b$ and $l - \tilde{l} > \tilde{m}_1 q + \tilde{m}_2 b$ one obtains with

$$l \;=\; \tilde{l} + l - \tilde{l} \;>\; \hat{m}_1 q + (\hat{m}_2 + 1)b + \tilde{m}_1 q + \tilde{m}_2 b \;=\; m_1 q + m_2 b$$

a contradiction to the assumption $l \leq m_1 q + m_2 b$. There are two a little more complicated cases left.

7. Assume that $\tilde{l} - b \leq (\hat{m}_1 - 1)q + (\hat{m}_2 - 1)b$ and $l - \tilde{l} > \tilde{m}_1 q + \tilde{m}_2 b$. Then there exists a $j \geq 1$ with

$$(\hat{m}_1 - j - 1)q + (\hat{m}_2 - j - 1)b \;<\; \tilde{l} - b \;\leq\; (\hat{m}_1 - j)q + (\hat{m}_2 - j)b\,.$$

Therefore, one has

$$\begin{aligned} l - \tilde{l} \;&>\; (m_1 - 1)q + (m_2 - 1)b - (\hat{m}_1 - j)q - (\hat{m}_2 - j + 1)b \\ &\geq\; (m_1 - \hat{m}_1 + j - 1)q + (m_2 - \hat{m}_2 + j - 2)b \\ &=\; (\tilde{m}_1 + j - 1)q + (\tilde{m}_2 + j - 1)b\,. \end{aligned}$$

Furthermore, it exists an $i \in \{1, \ldots, \beta(c-2, r-1)\}$ such that the identity $\hat{m}_1 = \phi(c, r-1) + i$ is valid. For $j < i$ it follows from the induction hypothesis that no less than

$$\hat{m}_1 - j - \beta(c, r-2) + m_1 - \hat{m}_1 + j - \beta(c-1, r-1) \;=\; m_1 - \beta(c, r-1)$$

checkpoint writings are needed to reverse the given l physical steps if states $\tilde{l} - q + 1, \ldots, \tilde{l}$ are copied to the second checkpoint. For $j \geq i$ one obtains that $j \leq i + \beta(c-1, r-2)$ because otherwise the number of physical steps performed to reverse $\tilde{l} - b$ physical steps would not be optimal. Hence, using the induction hypothesis one has that the minimal number of checkpoint writings needed to reverse the given l physical steps is no less than

$$\begin{aligned} &\beta(c-1, r-2) + m_1 - \hat{m}_1 + j - \beta(c-1, r-1) \;= \\ &\quad=\; \beta(c-1, r-2) + m_1 - \beta(c, r-2) - \beta(c-1, r-2) - i + j - \beta(c-1, r-1) \\ &\quad\geq\; m_1 - \beta(c, r-1)\,. \end{aligned}$$

8. Finally let $\tilde{l} - b > \hat{m}_1 q + \hat{m}_2 b$ and $l - \tilde{l} \leq (m_1 - \hat{m}_1 - 1)q + (m_2 - \hat{m}_2 - 2)b$ be valid. Then there exists a $j \geq 1$ fulfilling the inequality

$$(\hat{m}_1 + j - 1)q + (\hat{m}_2 + j - 1)b \;<\; \tilde{l} - b \;\leq\; (\hat{m}_1 + j)q + (\hat{m}_2 + j)b\,.$$

This yields

$$\begin{aligned} l - \tilde{l} \;&>\; (m_1 - 1)q + (m_2 - 1)b - (\hat{m}_1 + j)q - (\hat{m}_2 + j + 1)b \\ &\geq\; (m_1 - \hat{m}_1 - j - 1)q + (m_2 - \hat{m}_2 - j - 2)b \\ &=\; (\tilde{m}_1 + j - 1)q + (\tilde{m}_2 + j - 1)b\,. \end{aligned}$$

Consider $i \in \{0, \ldots, \beta(c-3, r)\}$ with $m_1 - \hat{m}_1 = \phi(c-1, r) + i$. For $j < i$ follows from the induction hypothesis that the minimal number of checkpoint writings is greater than or equal to

$$\hat{m}_1 + j - \beta(c, r-2) + m_1 - \hat{m}_1 - j - \beta(c-1, r-1) \;=\; m_1 - \beta(c, r-1)\,.$$

In the case $j \geq i$ one obtains from the induction hypothesis that at least

$$\begin{aligned}\hat{m}_1 + j - \beta(c, r-2) + \beta(c-2, r-1) \; &= \\ = \; & m_1 - \beta(c-1, r-1) - \beta(c-2, r-1) - i + j - \beta(c, r-2) + \beta(c-2, r-1) \\ \geq \; & m_1 - \beta(c, r-1)\end{aligned}$$

checkpoint writings are needed for the reversal of the given l physical steps.

Hence it has been shown for $l > \eta(c, r-1)$ that the minimal number of checkpoint writings needed to reverse the given l physical steps equals $s(l, c)$, which completes the proof. ∎

Figure 3.10 shows the principle structure of the function $s(l, c)$. It is, quite amazingly, piecewise constant, where the right margin belongs to the corresponding interval while the left margin does not.

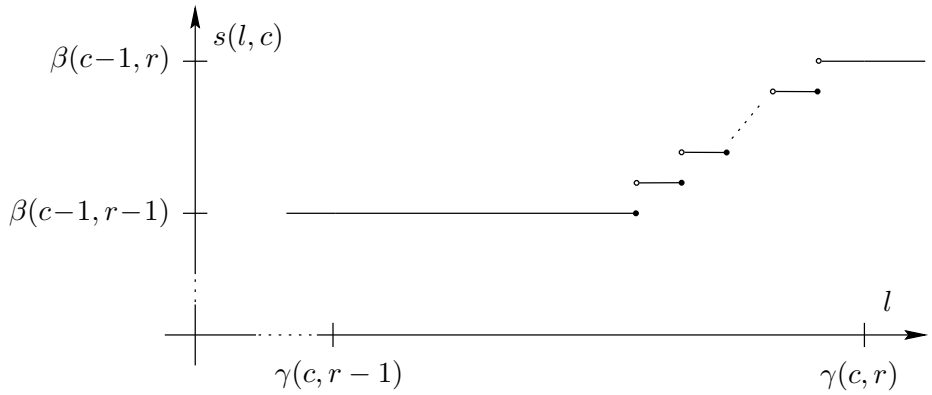


Figure 3.10: Optimal Number of Checkpoint Writings

3.3.2 Numerical Example

In this subsection the optimal serial reversal schedules developed in the previous subsection are applied to approximate the solution of the adjoint differential equation belonging to a particular nonlinear ordinary differential equation of order one. This problem is taken from [WNZD95]. The numerical results as well as the run times required are reported and discussed.

Model Problem

Consider the nonlinear ordinary differential equation

$$\frac{dz}{dt} = -z^2, \quad 0 < t < 1, \quad z(0) = u \in \mathbb{R} \; . \tag{3.14}$$

The analytic solution of this differential equation is given by

$$z(t) = \frac{u}{tu + 1} \; . \tag{3.15}$$

Suppose that observations z^* of the model are generated with the initial condition $z^*(0) = 1$. Then one has for the observation z^* that

$$z^*(t) = \frac{1}{t+1} \; . \tag{3.16}$$

One may define the cost function

$$\begin{aligned} J(u) &= \frac{1}{2}\int_0^1 < z - z^*,\, z - z^* > dt \\ &= \frac{1}{2}\int_0^1 \left(\frac{u}{tu+1} - \frac{1}{t+1}\right)^2 dt \\ &= \frac{1}{2}\left(1 + u + \frac{2u}{1-u}\ln\left(\frac{u+1}{2}\right) - \frac{1+3u}{2(u+1)}\right) \; . \end{aligned} \tag{3.17}$$

The first order adjoint equation of system (3.14) equals

$$-\frac{\partial \bar{z}}{\partial t} \;=\; -2z\bar{z} + z - z^*, \quad \bar{z}(1) \;=\; 0 \; , \tag{3.18}$$

where $\bar{z}$ is the adjoint variable [WNZD95]. The gradient of the cost function with respect to the initial conditions is given by $\nabla_u J = \bar{z}(0)$ [WNZD95]. Hence the gradient is known if the solution of the adjoint system (3.18) is available. For the model problem considered $\bar{z}$ has the analytic solution [WNZD95]

$$\begin{aligned} \bar{z}(t) &= (tu+1)^2\left(\int_0^1 \left(\frac{1}{t+1} - \frac{u}{tu+1}\right)\frac{1}{(tu+1)^2}dt + C\right) \\ &= (tu+1)^2\left(\frac{1}{(1-u)(tu+1)}\right. \\ &\quad \left. + \frac{1}{(1-u)^2}\ln\left(\frac{t+1}{ut+1}\right) + \frac{1}{2(tu+1)^2} + C\right), \end{aligned} \tag{3.19}$$

where C is the constant

$$C = -\frac{1}{1-u^2} - \frac{1}{(1-u)^2}\ln\left(\frac{2}{u+1}\right) - \frac{1}{2(u+1)^2}$$

determined by the final condition $\bar{z}(1) = 0$.

Three different multi-step evolutions each of which calculates an approximation of z are considered. The reversal of F is applied to evaluate an approximation of $\bar{z}$. Thus one can check the correctness of $\bar{z}$ besides the analysis of the run-time behavior. For the three multi-step evolutions the value of J is approximated using the integration

$$J \;\; += \;\; \frac{h}{2}\left(z_j - z_j^*\right)^2 \;=\; \frac{h}{2}\left(z_j - 1/(hj+1)\right)^2 \; . \tag{3.20}$$

As discretization schemes the leap-frog method, the explicit Adams 3-step method, and a Milne 4-step method are applied (see e.g. [SB90], [HNW96]). The explicit Euler method forms the start-up calculation.

For the approximation of z applying the leap-frog method with the time step size h and the right-hand side $f(z) = z^2$ of the ordinary differential equation under consideration, i.e.

$$z_{j+1} = z_{j-1} + 2hf(z_j) ,$$

one can determine the adjoint of the 2-step method according to the procedure described in [Gri00]. This yields with $\bar{J} = 1$

$$\bar{z}_{j-1} = \bar{z}_{j+1} + 2h\bar{z}_j f'(z_{j-1}) + h(z_{j-1} - 1/(h(j-1)+1)) \qquad l \geq j > 1 .$$

For the adjoint of the discretization method the last term on the right-hand side takes the computation of J into account (see Equation (3.20)). The corresponding 2-step evolution F was already derived in Example 2.2 of Chapter 2. The same expression for $\bar{z}_{j-1}$ using $f'(z_j)$ instead of $f'(z_{j-1})$ represents the discretization of the adjoint differential equation applying the leap-frog method.

If the Adams 3-step method is used to calculate an approximation of z, i.e.

$$z_{j+1} = z_j + h\left(\frac{23}{12}f(z_j) - \frac{16}{12}f(z_{j-1}) + \frac{5}{12}f(z_{j-2})\right)$$

with the time step size h, the corresponding adjoint of this 3-step method is given by

$$\bar{z}_{j-1} = \bar{z}_j + h\left(\frac{23}{12}\bar{z}_j - \frac{16}{12}\bar{z}_{j+1} + \frac{5}{12}\bar{z}_{j+2}\right) f'(z_{j-1}) + hz_{j-1} - \frac{h}{h(j-1)+1} .$$

One can define the physical steps F_i of an appropriate multi-step evolution F as

$$x_{i+1} = (y_i, z_{i+1}) \equiv \left(f(z_i), z_i + \frac{23h}{12}y_i - \frac{16h}{12}y_{i-1} + \frac{5h}{12}y_{i-2}\right) \equiv F_i(x_{i-1}, x_i) .$$

Hence, in this case F equals a 2-step evolution. Furthermore it is possible to observe that F_i depends nonlinear only on z_i. This yields that the number b of linear arguments equals 1. The reverse steps $\bar{F}_i$ are defined as

$$\begin{aligned} \bar{x}_{i-1} &= (\bar{y}_{i-2}, \bar{z}_{i-1}) \mathrel{+}= \left(\frac{5h}{12}\bar{z}_{i+1}, 0\right), \\ \bar{x}_i &= (\bar{y}_{i-1}, \bar{z}_i) \\ &\mathrel{+}= \left(-\frac{16h}{12}\bar{z}_{i+1}, \bar{z}_{i+1} + \left(\frac{23h}{12}\bar{z}_{i+1} + \bar{y}_i\right) f'(z_i) + hz_i - \frac{h}{hi+1}\right) . \end{aligned}$$

Suppose that $\bar{x}_i$ is known. During the release of two checkpoints that comprise the successive state vectors $i-1$ and i one can perform a recording step that stores the intermediates calculated during the evaluation of f at the argument z_{i-1} on the tape. Therefore it is possible to evaluate $\bar{F}_{i-1}$ since all needed values are available. It follows again that b is equal to 1 for this 2-step evolution F.

Finally, if the Milne 4-step method

$$z_{j+1} = z_{j-3} + h\left(\frac{8}{3}f(z_j) - \frac{4}{3}f(z_{j-1}) + \frac{8}{3}f(z_{j-2})\right) ,$$

with the time step size h, is used to approximate z, the corresponding adjoint of this 4-step method equals

$$\bar{z}_{j-1} \;=\; \bar{z}_{j+3} + h\left(\frac{8}{3}\bar{z}_j - \frac{4}{3}\bar{z}_{j+1} + \frac{8}{3}\bar{z}_{j+2}\right) f'(z_{j-1}) + hz_{j-1} - \frac{h}{h(j-1)+1}$$

In order to define an appropriate multi-step evolution F one may use the physical steps F_i

$$\begin{aligned} x_{i+1} \;=\; (y_i, z_{i+1}) \;&\equiv\; \left(f(z_i), z_{i-3} + \frac{8h}{3}y_i - \frac{4h}{3}y_{i-1} + \frac{8h}{3}y_{i-2}\right) \\ &\equiv\; F_i(x_{i-3}, x_{i-2}, x_{i-1}, x_i) \end{aligned}$$

and the reverse steps $\bar{F}_i$ with

$$\begin{aligned} \bar{x}_{i-3} \;&=\; (\bar{y}_{i-4}, \bar{z}_{i-3}) \;+\!=\; (0, \bar{z}_{i+1})\,, \\ \bar{x}_{i-2} \;&=\; (\bar{y}_{i-3}, \bar{z}_{i-2}) \;+\!=\; (0, 0)\,, \\ \bar{x}_{i-1} \;&=\; (\bar{y}_{i-2}, \bar{z}_{i-1}) \;+\!=\; \left(\frac{8h}{3}\bar{z}_{i+1}, 0\right), \\ \bar{x}_i \;&=\; (\bar{y}_{i-1}, \bar{z}_i) \quad +\!=\; \left(-\frac{4h}{3}\bar{z}_{i+1}, \left(\frac{8h}{3}\bar{z}_{i+1} + \bar{y}_i\right) f'(z_i) + hz_i - \frac{h}{hi+1}\right). \end{aligned}$$

A similar argument as above yields $b = 3$ for this 4-step evolution.

The three multi-step evolutions are used together with the steering routine described in next subsection to calculate approximations of z and $\bar{z}$.

Available Software

The C-routine `revolve` a code listing of which is contained in Appendix A implements the optimal serial schedules derived in Section 3.3.1. Depending on the actual number of physical steps l to be reversed and the number c of checkpoints that are available at the moment `revolve` selects the next states $\hat{l} - q + 1, \ldots, \hat{l}$ to be copied to a checkpoint according to

$$\hat{l} = \begin{cases} \gamma(c, r-1)+b & \text{if } l \le \gamma(c, r-1)+\beta(c-2, r-1)q \\ l-\phi(c-1, r)q-(\beta(c-2, r-1)-1)b & \text{else} \end{cases}$$

if one has $l \le \phi(c, r)q + (\beta(c-1, r-1) - 1)b$. Otherwise, i.e. if the inequality $l > \phi(c, r)q + (\beta(c-1, r-1) - 1)b$ is valid, $\hat{l}$ is chosen according to

$$\hat{l} = \begin{cases} (m_1-\beta(c-1, r))(q+b)-(\beta(c, r-2)+1)b & \text{if } \phi(c, r)+\beta(c-3, r) < m_1 \\ (\phi(c, r-1)+1)q+(\beta(c-1, r-2)-1)b & \text{else} \end{cases}$$

with m_1 as defined above. Because of the proofs given in Section 3.3.1 this choice guarantees that the minimal number $t(l, c)$ of physical steps and the minimal number $s(l, c)$ of checkpoint writings are performed during the reversal of l physical steps.

To reverse a given sequence of physical steps `revolve` can be thought of as a "controller" steering the reversal process. To apply `revolve` one has to provide procedures for the tasks

(a) Advance F to a certain state (`forward(..)`)

(b) Perform at most q recording steps and reverse steps (with special provisions for the first time this happens) (`forwardrec(..)`, `reverse(..)`)

(c) Copy the current states to a checkpoint (`store_checkpoint(..)`)

(d) Read the most recently saved program states (`restore_checkpoint(..)`)

These tasks represent the actions defined already in Def. 2.3, that introduced serial reversal schedules. Having them at hand the reversal of the evolutionary system can be performed in principle according to the following code segment. It is contained within the program to calculate $\bar{z}$ as solution of the adjoint equation (3.18), where `revolve` determines the task to execute:

```
...  /* declarations and initializations */
do
 { oldcapo = capo;
   whatodo = revolve(&check,&capo,&fine,snaps,q,b,&num,&info);
   switch(whatodo)
    {   /* Advancing F to a certain state */
     case advance:  for(j=oldcapo;j<capo;j++)
                      forward(..);
                    break;
        /* Save the current states onto a stack */
     case takeshot: store_checkpoint(..);
                    break;
        /* Restore the most recently saved checkpoint */
     case restore:  restore_checkpoint(..);
                    break;
        /* Perform num recording steps and reverse steps */
     case firsturn: forwardrec(..,num);
                    init_x_bar(..);
                    reverse(..,num);
                    break;
        /* Perform num recording steps and reverse steps */
     case youturn:  forwardrec(..,num);
                    reverse(..,num);
                    break;
        /* Perform b reverse steps during release */
     case reduce_checkpoint:
                    forwardrec(..,b);
                    reverse(..,b);
                    break;
     case error:    printf("irregular termination of revolve\n");
    }
 } while((whatodo != terminate) && (whatodo != error));
```

Numerical Results and Run Times

Approximations of $\bar{z}$ were computed using the three different multi-step evolutions on a Pentium PC. Figure 3.11 shows the relative error of $\bar{z}$ computed with the adjoint of the Adams 3-step method in comparison with the analytic solution $\bar{z}$ given by Equation (3.19) for $u = 1.5$ and five different numbers of physical steps starting with $l = 1000$ (i.e. $h = 0.001$) up to $l = 5000$. For

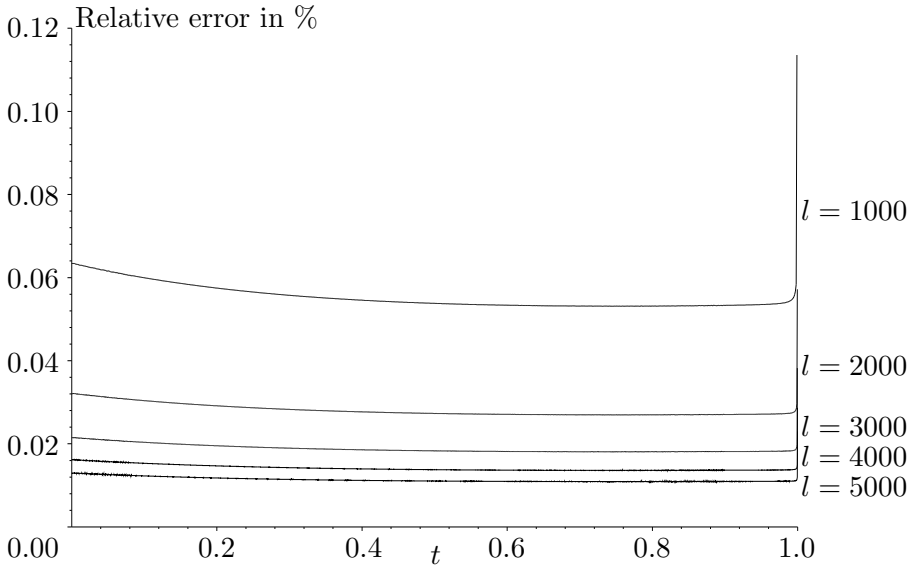


Figure 3.11: Relative Error of $\bar{z}$ Using Adjoint of the Adams 3-step Method

$t = 1$ the error is limited above by 0.12 % for the five values of l. The principal behavior of the approximations computed with the leap-frog method and the Milne 4-step method, respectively, is more or less the same.

As mentioned already in Chapter 2 the reverse steps $\bar{F}_i$ represents the adjoint of the discretization method described by F. Naturally, as alternative one could use the discretized adjoint differential equation to approximate $\bar{z}$. For example then one obtains for the leap-frog method

$$\bar{z}_i \;\equiv\; \bar{z}_{i+2} + 2h\bar{z}_{i+1}f'(z_{i+1}) \; .$$

Hence the difference is given by the fact that $\bar{F}_i$ utilizes $f'(z_i)$ to calculate $\bar{z}_i$ whereas in the discretized adjoint differential equation $f'(z_{i+1})$ is used to calculate $\bar{z}_i$. The approximation calculated with the discretized adjoint differential equation and the approximation calculated with the reverse steps $\bar{F}_i$ are almost identical. Furthermore the relative error of $\bar{z}$ computed with the discretized adjoint differential equation has the same order and shows the same behavior as the relative error of $\bar{z}$ computed with the reverse steps $\bar{F}_i$. Therefore it make sense to utilize the adjoints $\bar{F}_i$ of the physical steps F_i. This behavior can be observed for all discretization schemes considered so far.

Figure 3.12 illustrates the run-time behavior for the reversal of 10000 physical steps using the routine `revolve`. The leap-frog method was applied as

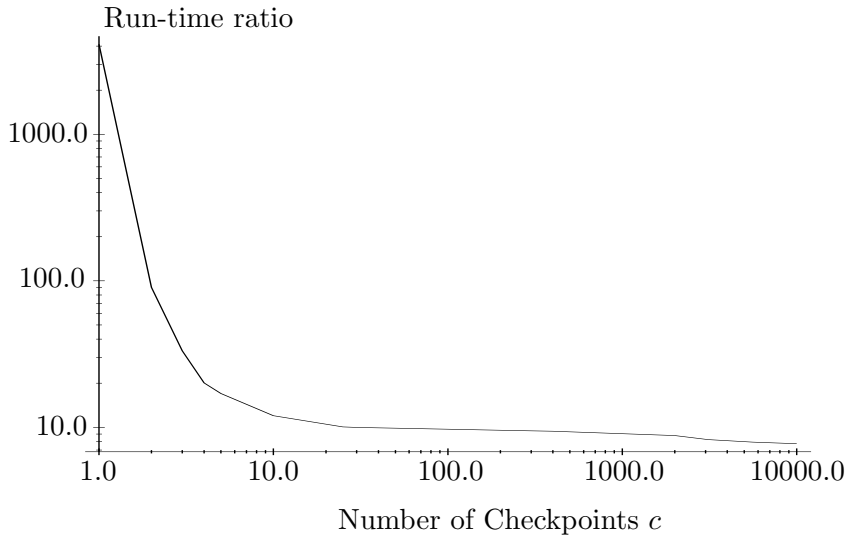


Figure 3.12: Run-time Ratio for Different Numbers of Checkpoints

discretization scheme. For both axes a logarithmic scale is utilized. The run time of the reversal is normalized by the run time required to evaluate F and displayed as function of the number of checkpoints c that are available. Therefore, the value $c = 10000$ represents the memory requirement of the basic approach where the complete execution log is recorded once. Here, the ratio 7.7 is near the theoretical minimum of 5 for the ratio between the run time of the reversal and of the function evaluation [Gri89], where only a flat memory hierarchy is assumed. As one can see it is possible to push the number of checkpoints that are available far below 100, i.e. below 1 % of the number of physical steps to be reversed, before a significant impact on the temporal complexity becomes apparent. Only when the number of checkpoints becomes lower than 10 the run time increases drastically. Nevertheless these thresholds are not constant and will decline as the total number of physical steps to be reversed grows.

3.3.3 Conclusions

Assume the given evolutionary system F applies a one-step recurrence such that no reverse step can be performed during the release of a checkpoint, i.e. $q = 1$ and $b = 0$. Then one obtains for the minimal number $t(l, c)$ of physical steps that have to be performed and the minimal number $s(l, c)$ of checkpoint writings during the reversal exactly the formulas already derived in [GW00]. Hence the new results presented in this thesis can be seen as a generalization of the known theory for uniform step costs.

Furthermore, for the integer r determined by the given number l of physical steps and the number of checkpoints c that are utilized Theorem 3.1 yields the following inequality

$$(r-1)\frac{c}{c+1} \leq \frac{t(l,c)}{l} \leq r \ .$$

Hence, if c is of significant size the value of r can be considered as integer

approximation to the ratio between the cost of reversing the given l physical steps and the cost of evaluating the evolutionary system F formed by the l physical steps. More precisely it can be shown that the integer r denotes the maximal number of times a particular physical step is performed during the reversal of F.

Rather than achieving the minimal number of physical steps $t(l, c)$ performed for given l and c one may ask for the behavior of the maximal number l of physical steps that can be reversed with up to c checkpoints and a given value of r. Using the assumption of Theorem 3.1

$$\beta(c, r-1)q + (\beta(c-1, r-1) - 1)b < l \leq \beta(c, r)q + (\beta(c-1, r) - 1)b$$

with q and b determined by the multi-step evolution under consideration for the maximal number $l(c, r)$ of physical steps that can be reversed follows

$$\begin{aligned}
l(c,r) &= \beta(c,r)q + (\beta(c-1,r)-1)b = \binom{c+r}{c} q + \left(\binom{c+r-1}{c-1} - 1\right) b \\
&= \frac{(c+r)!\,q}{c!\,r!} + \left(\frac{(c+r-1)!}{(c-1)!\,r!} - 1\right) b \\
&\approx \left(\frac{1}{\sqrt{2\pi}} \left(1 + \frac{r}{c}\right)^c \left(1 + \frac{c}{r}\right)^r \sqrt{\frac{1}{c} + \frac{1}{r}}\right) q \\
&\quad + \left(\frac{1}{\sqrt{2\pi}} \left(1 + \frac{r}{c-1}\right)^{c-1} \left(1 + \frac{c-1}{r}\right)^r \sqrt{\frac{1}{c-1} + \frac{1}{r} - 1}\right) b \\
&\sim \frac{q}{\sqrt{2\pi \min(c,r)}} \left(\frac{\max(c,r)}{\min(c,r)}\,\mathrm{e}\right)^{\min(c,r)} \\
&\quad + \frac{b}{\sqrt{2\pi \min(c-1,r)}} \left(\frac{\max(c-1,r)}{\min(c-1,r)}\,\mathrm{e}\right)^{\min(c-1,r)} .
\end{aligned}$$

The first approximation is derived using Stirling's formula. From the last formula one obtains

$$l(c,r) \sim \begin{cases} (q + b/\mathrm{e})\exp(r)/\sqrt{r} & \text{if } c \sim r \\ q\,c^r + b\,(c-1)^r & \text{if } c \gg r = const \\ (q + b/r)r^c & \text{if } r \gg c = const \end{cases} .$$

Hence, if one accepts a similar growth in the number of checkpoints that are used and the increase in run time, i.e. $c \sim r$, one obtains a logarithmic growth in the memory requirement and in the run time for the reversal of an evolutionary system. Suppose that $q = 2$ and $b = 1$ (e.g. leap-frog method). If one has $c = r = 8$, i.e. $cq = 16$ states can be copied to checkpoints, the maximal number of physical steps that can be reversed equals 32 174. Assume that for $q = 4$ and $b = 3$ (e.g. Milne 4-step method) the equality $c = r = 8$ is valid. Hence, $cq = 32$ states can be copied to checkpoints. Then one finds $l(c, r) = 70\,782$. For the usual case, where one has a fixed amount of memory, i.e. c is constant and r increases, one finds that $l(c, r)$ grows more or less like the cth power of r.

Summarizing the above one can conclude that for $b > 0$ some gain can be achieved applying the new optimal serial reversal schedules for multi-step recurrences instead of using the mega-step approach described in the beginning of this section.

3.4 One-step Evolutions with Non-uniform Step Costs

For the remainder of this chapter only one-step evolutions F combining l physical steps will be considered. It is supposed that the physical steps have non-uniform step costs and $b = 0$ linear arguments as defined in Def. 2.1. The derived results can be extended easily to the case $q > 1$ and $b = 0$, where the mega-step approach described in the beginning of Section 3.3 can be applied.

3.4.1 Monotony of Partitioning

As seen already in Example 3.1 the optimal serial reversal schedules derived in Section 3.3 for uniform step costs do not usually lead to the minimal cost $t(0, l, c)$ in the non-uniform case. Nevertheless because of its meaning described above the integer r denotes an approximation of the complexity rate $t(0, l, c)/l$. To find an optimal serial reversal schedule for non-uniform step costs a search algorithm has to be used. The technique of dynamic programming [AHU74] provides a possibility to find with temporal complexity $\mathcal{O}(cl^3)$ a serial reversal schedule S that attains $t(0, l, c)$. I.e. one can determine S for reversing $l > 1$ physical steps with non-uniform step costs using up to $c > 1$ checkpoints and

$$t_S(0, l, c) = t(0, l, c) = \min_{1 \le \tilde{l} < l} \left\{ \sum_{j=0}^{\tilde{l}-1} t_j + t(\tilde{l}, l, c-1) + t(0, \tilde{l}, c) \right\}$$

in a run time the behavior of which is cubic in l. For example one may apply the following algorithm to examine all possible serial reversal schedules:

```
for(s=1;s<=c;s++)                          /* for all checkpoint */
{ for(d=2;d<=l;d++)                 /* for all possible distances */
    for(i=0;i<=l-d;i++)                     /* for all possible i */
    { h = i+d;
      if (d <= s+1)   ...                /* some initializations */
      else
      { if (s == 1)   ...                /* some initializations */
        else
        { lb_m = i+1;
          ub_m = h-1;
          part[s][i+h*(h-1)/2] = 0; /* part = next checkpoint */
          t[s][i+h*(h-1)/2] = -1;
          for(m=lb_m;m<=ub_m;m++)
          {   /* tenp = ti[i]+...+ti[m-1]+t(m,h,s-1)+t(i,m,s) */
            if (i > 0)
```

```
              tenp = sum[m-1]-sum[i-1];
            else
              tenp = sum[m-1];
            tenp += t[s-1][m+h*(h-1)/2]+t[s][i+m*(m-1)/2];
            if((tenp<t[s][i+h*(h-1)/2])||(t[s][i+h*(h-1)/2]==-1))
            { t[s][i+h*(h-1)/2] = tenp;
              part[s][i+h*(h-1)/2] = m;
}   } } } } }
```

which has the stated complexity because of the 4 nested `for`-loops. The array `part` stores the information on which state has to be copied to the next checkpoint. The array `t` denotes the values of $t(i,h,c)$. In order to reduce the run time needed by this algorithm to find an optimal serial reversal schedule one may apply a property of the checkpoint writings proven in the remainder of this subsection. For that purpose the partition of the given physical steps into two sequences that are reversed separately is studied in detail.

Definition 3.3 (Minimal Partition Function). Assume that the given sequence of l physical steps has the step costs t_j, $0 \le j < l$. Let $c > 1$ checkpoints be available. The *minimal partition function* $P(i,h,c)$ denotes for $0 < i+1 < h \le l$ the minimal state that can be copied to the second checkpoint such that the minimal cost $t(i,h,c)$ is attained, i.e.

$$P(i,h,c) \;\equiv\; \min\left\{ i < m < h \;\middle|\; \sum_{j=i}^{m-1} t_j + t(m,h,c-1) + t(i,m,c) = t(i,h,c) \right\} .$$

Theorem 3.3 (Monotonic Partitioning).
Suppose a sequence of l physical steps with the step costs t_j, $0 \le j < l$, is given. Let $c > 1$ checkpoints be available. For $0 < i+1 < h \le l$ the minimal partition function $P(i,h,c)$ is monotonic in i and h, respectively.

Proof. For arbitrary i and h with $1 < i+1 < h < l$ define

$$m_1 \;\equiv\; P(i-1,h,c), \qquad m_2 \;\equiv\; P(i,h,c), \qquad m_3 \;\equiv\; P(i,h+1,c)\,. \quad (3.21)$$

In order to prove the assertion one has to show that $m_1 \le m_2 \le m_3$. To do so an induction on c will be used:

$c = 2$:
First, $m_2 \le m_3$ will be shown. For that the following property of $t(i,h,1)$, as a consequence of Lemma 3.2, is needed:

$$\begin{aligned} & t(i-1,h+1,1) - t(i,h+1,1) - (t(i-1,h,1) - t(i,h,1)) \;= \\ &= \sum_{j=i-1}^{h} (h-j)t_j - \sum_{j=i}^{h}(h-j)t_j - \left(\sum_{j=i-1}^{h-1} (h-j-1)t_j - \sum_{j=i}^{h-1}(h-j-1)t_j \right) \\ &= (h-i+1)t_{i-1} - (h-i)t_{i-1} \;=\; t_{i-1} \;>\; 0\,. \end{aligned}$$

This provides the inequality

$$t(i-1,h+1,1) - t(i-1,h,1) \;\;\ge\;\; t(i,h+1,1) - t(i,h,1)\,. \quad (3.22)$$

Assume that $m_2 > m_3$ and consider

$$\begin{aligned} \tilde{t}(i,h,2) &= \sum_{j=i}^{m_3-1} t_j + t(m_3,h,1) + t(i,m_3,2) \quad \text{and} \\ \tilde{t}(i,h+1,2) &= \sum_{j=i}^{m_2-1} t_j + t(m_2,h+1,1) + t(i,m_2,2)\ . \end{aligned}$$

The fact $t(i,h+1,2) \leq \tilde{t}(i,h+1,2)$ yields

$$t(m_3,h+1,1) - t(m_2,h+1,1) \leq \sum_{j=m_3}^{m_2-1} t_j + t(i,m_2,2) - t(i,m_3,2)\ .$$

Using Inequality (3.22) one obtains

$$t(m_3,h,1) - t(m_2,h,1) \leq t(m_3,h+1,1) - t(m_2,h+1,1)\ .$$

The last two inequalities can be combined to derive

$$t(m_3,h,1) - t(m_2,h,1) \leq \sum_{j=m_3}^{m_2-1} t_j + t(i,m_2,2) - t(i,m_3,2)\ ,$$

which implies $\tilde{t}(i,h,2) \leq t(i,h,2)$. This is a contradiction to the definition of m_2 in Def. (3.21). Hence, $m_2 \leq m_3$ must be valid.

To show $m_1 \leq m_2$ a second induction on $h-i$ is used. For $h-i=2$ it is easy to check that the assertion is true and that the inequality

$$t(m-1,h-1,2) - t(m,h-1,2) \leq t(m-1,h,2) - t(m,h,2) \quad (3.23)$$

is valid for $i \leq m \leq h-1$, i.e. for $m=i$ or $m=i+1$. Suppose that Inequality (3.23) holds for $\tilde{h} - \tilde{\imath} \leq M-1$ and consider $h-i=M$. Defining

$$\begin{aligned} \bar{t}(i-1,h,2) &\equiv \sum_{j=i-1}^{m_2-1} t_j + t(m_2,h,1) + t(i-1,m_2,2) \quad \text{as well as} \\ \bar{t}(i,h,2) &\equiv \sum_{j=i}^{m_1-1} t_j + t(m_1,h,1) + t(i,m_1,2) \end{aligned}$$

and assuming $m_2 < m_1$ it follows that $t(i,h,2) \leq \bar{t}(i,h,2)$ because of the minimum given by $t(i,h,2)$. Therefore one obtains

$$t(i,m_2,2) - t(i,m_1,2) \leq \sum_{j=m_2}^{m_1-1} t_j + t(m_1,h,1) - t(m_2,h,1)\ .$$

For $m_2 < m_1 \leq h-1$ the Inequality (3.23) holds because of the induction hypothesis. Hence one has

$$t(i-1,m_2,2) - t(i-1,m_1,2) \leq t(i,m_2,2) - t(i,m_1,2)\ .$$

The last two inequalities yield

$$t(i-1,m_2,2)-t(i-1,m_1,2) \;\leq\; \sum_{j=m_2}^{m_1-1} t_j + t(m_1,h,1) - t(m_2,h,1)$$

and one finds $\bar{t}(i-1,h,2) \leq t(i-1,h,2)$. This is a contradiction to the definition of m_1 in Def. (3.21). Therefore $m_1 \leq m_2$ must be valid and the assertion is proven for $h-i=M>2$. In order to perform the next step of the induction on $h-i$ it is left to show that for all $i \leq m \leq h-1$

$$t(m-1,h-1,2)-t(m,h-1,2) \;\leq\; t(m-1,h,2)-t(m,h,2) \quad (3.24)$$

holds. For m with $i+1 \leq m \leq h-1$ Inequality (3.24) is valid because of $h-m \leq h-(i+1) = M-1$ and the induction hypothesis that provides Inequality (3.23). Hence, it is left to show that Inequality (3.24) is valid for $m=i$. With m_4 and m_5 defined as

$$m_4 \;\equiv\; P(i-1,h-1,2) \qquad \text{and} \qquad m_5 \;\equiv\; P(i,h-1,2)$$

the induction hypothesis and the properties shown so far yield the relations $m_4 \leq m_1 \leq m_2$ and $m_4 \leq m_5 \leq m_2$. In the case $m_1 \leq m_5$ the inequalities $t(m_1,h-1,1)-t(m_5,h-1,1) \leq t(m_1,h,1)-t(m_5,h,1)$ and

$$t(i-1,h-1,2) \;\leq\; \sum_{j=i-1}^{m_1-1} t_j + t(m_1,h-1,1) + t(i-1,m_1,2)$$

hold. The last inequality is valid because $t(i-1,h-1,2)$ denotes the minimal cost to reverse the physical steps between $i-1$ and $h-1$. Therefore, it is possible to conclude that

$$\begin{aligned}
&t(i-1,h-1,2)-t(i,h-1,2) \;\leq \\
&\leq \sum_{j=i-1}^{m_1-1} t_j + t(m_1,h-1,1) + t(i-1,m_1,2) - \sum_{j=i}^{m_5-1} t_j - t(m_5,h-1,1) - t(i,m_5,2) \\
&\leq \sum_{j=i-1}^{m_1-1} t_j + t(m_1,h,1) + t(i-1,m_1,2) - \sum_{j=i}^{m_5-1} t_j - t(m_5,h,1) - t(i,m_5,2) \\
&\leq\; t(i-1,h,2) - t(i,h,2)
\end{aligned}$$

and Inequality (3.24) is proven for $m=i$. For $m_5 < m_1$ assume that the strict inequality

$$t(i-1,h-1,2)-t(i,h-1,2) \;>\; t(i-1,h,2)-t(i,h,2)$$

is valid. This yields

$$\begin{aligned}
&\sum_{j=m_4}^{m_1-1} t_j + t(m_1,h,1) + t(i-1,m_1,2) - t(m_4,h-1,1) - t(i-1,m_4,2) \;< \\
&\qquad < \sum_{j=m_5}^{m_2-1} t_j + t(m_2,h,1) + t(i,m_2,2) - t(m_5,h-1,1) - t(i,m_5,2)\,.
\end{aligned}$$

Furthermore, one has $t(i,m_1,2)-t(i-1,m_1,2) \leq t(i,m_5,2)-t(i-1,m_5,2)$ because of $m_5 < m_1 \leq h-1$ and the induction hypothesis for $m_1 - i \leq M-1$. Using these inequalities and the minimization property of m_4 one finds by adding the term $t(i,m_1,2)-t(i,m_1,2)$ on the right-hand side

$$\begin{aligned}
& t(m_1,h,1)-t(m_2,h,1) \; < \\
< \; & \sum_{j=m_1}^{m_2-1} t_j + t(i,m_2,2) - t(i,m_1,2) + t(i,m_5,2) - t(i,m_5,2) - \sum_{j=m_4}^{m_5-1} t_j \\
& - t(m_5,h-1,1) - t(i-1,m_5,2) + t(m_4,h-1,1) + t(i-1,m_4,2) \\
= \; & \sum_{j=m_1}^{m_2-1} t_j + t(i,m_2,2) - t(i,m_1,2) + t(i-1,h-1,2) - \sum_{j=i-1}^{m_5-1} t_j \\
& - t(i-1,m_5,2) - t(m_5,h-1,1) \\
\leq \; & \sum_{j=m_1}^{m_2-1} t_j + t(i,m_2,2) - t(i,m_1,2)
\end{aligned}$$

because $t(i-1,h-1,2)$ is minimal and hence

$$\sum_{j=i}^{m_1-1} t_j + t(m_1,h,1) + t(i,m_1,2) \;<\; \sum_{j=i}^{m_2-1} t_j + t(m_2,h,1) + t(i,m_2,2)\,.$$

This implies that the cost for reversing the physical steps between state i and state h using m_1 rather than m_2 is lower than $t(i,h,2)$. This is a contradiction to the choice of m_2 in Equation (3.21). Hence also in the case $m_5 < m_1$ the inequality $t(i-1,h-1,2)-t(i,h-1,2) \leq t(i-1,h,2)-t(i,h,2)$ is valid and therefore Inequality (3.24) is proven for $m = i$.

Now time has come to draw conclusions from the results shown so far. Starting with the inequality

$$t(i,h+1,c-1)-t(i,h,c-1) \;\leq\; t(i-1,h+1,c-1)-t(i-1,h,c-1)$$

for $c = 2$ it was shown that $m_2 \leq m_3$. This was applied in a second induction to prove $m_1 \leq m_2$ and

$$t(i,h+1,c)-t(i,h,c) \;\leq\; t(i-1,h+1,c)-t(i-1,h,c)\,.$$

Exact the same argument as above can be used to perform the induction step from $c-1$ to c if $c > 2$, which completes the proof. ∎

The conclusion that can be drawn from Theorem 3.3 is the following. Consider the physical steps between state i and state h as well as $m \equiv P(i,h,c)$. Hence m equals the minimal partition of the physical steps between state i and state h such that the minimal cost $t(i,h,c)$ for the reversal is attained. Because the minimal partition function p is monotonic, the range of possibilities for the determination of m can be reduced from $\{i < m < h-1\}$ to $\{m_l \leq m < h-1\}$ with $m_l \equiv P(i,h-1,c)$. Using $m \leq m_u$ with $m_u \equiv P(i+1,h,c)$ it is possible to reduce the search interval further to $\{m_l \leq m \leq m_u\}$. These reductions can be used quite efficiently in the algorithm to find one serial reversal schedule needing the minimal cost $t(i,h,c)$ as shown in the next subsection.

3.4.2 Numerical Examples

Software Available

The routine shown at Page 48 was modified employing the results of the last theorem. The full source code of this improved algorithm can be found in Appendix A. Because of the `for`-loop over d the lower and upper bound of m are always available such that the search can be reduced correspondingly.

```
for(s=1;s<=c;s++)                        /* for all checkpoint */
{ for(d=2;d<=l;d++)                 /* for all possible distances */
    for(i=0;i<=l-d;i++)                  /* for all possible i */
    { h = i+d;
      if (d <= s+1)
        ...                        /* initializations as above */
      else
      { if (s == 1)
          ...                      /* initializations as above */
        else
        {                                 /* the only changes: */
          lb_j = part[s][i+(h-1)*(h-2)/2];
          ub_j = part[s][i+1+h*(h-1)/2];
           /* instead of   lb_m = i+1;   and   ub_m = h-1; */
                                 /* part = next checkpoint */
          part[s][i+h*(h-1)/2] = 0;
          t[s][i+h*(h-1)/2] = -1;
          for(m=lb_m;m<=ub_m;m++)
          {
           /* tenp = ti[i]+...+ti[m-1]+t(m,h,s-1)+t(i,m,s) */
            if (i > 0)
              tenp = sum[m-1]-sum[i-1];
            else
              tenp = sum[m-1];
            tenp += t[s-1][m+h*(h-1)/2]+t[s][i+m*(m-1)/2];
            if((tenp<t[s][i+h*(h-1)/2])||(t[s][i+h*(h-1)/2]==-1))
            { t[s][i+h*(h-1)/2] = tenp;
              part[s][i+h*(h-1)/2] = m;
}   } } } } }
```

The behavior of the memory requirement of the modified algorithm as well as of the original algorithm is equal to $\mathcal{O}(cl^2)$ because for almost all $1 \leq s \leq c$ and $0 \leq i < h \leq l$ the cost $t(i, h, s)$ is stored in the array `t`.

Both algorithms were applied to calculate optimal serial reversal schedules for different kinds of one-step evolutions with non-uniform step costs. The resulting improvements using the modified algorithm with respect to a decreased run time are reported next.

Run-time Tests

In order to test the modified search algorithm four different distributions of the step costs t_i, $0 \leq i < l$, are considered, namely linearly increasing step costs with $t_i = 0.05(i+1)$, linearly decreasing step costs, i.e. $t_i = l - 0.05(i+1)$, step costs with a Gaussian distribution $t_i = l - \exp(-0.5(4i/l - 2)^2)$, and random distributed step costs $t_i = \text{rand}()$. For each of these four possibilities and $c = 5, 10, 20$ the CPU time t_o needed by the original algorithm and the CPU time t_m required by the modified algorithm to calculate an optimal serial reversal schedule are measured. The run times required for the different kinds of non-uniform step costs are almost the same. Hence, Fig. 3.13 displays the results achieved only for linearly increasing step costs. A logarithmic scaling

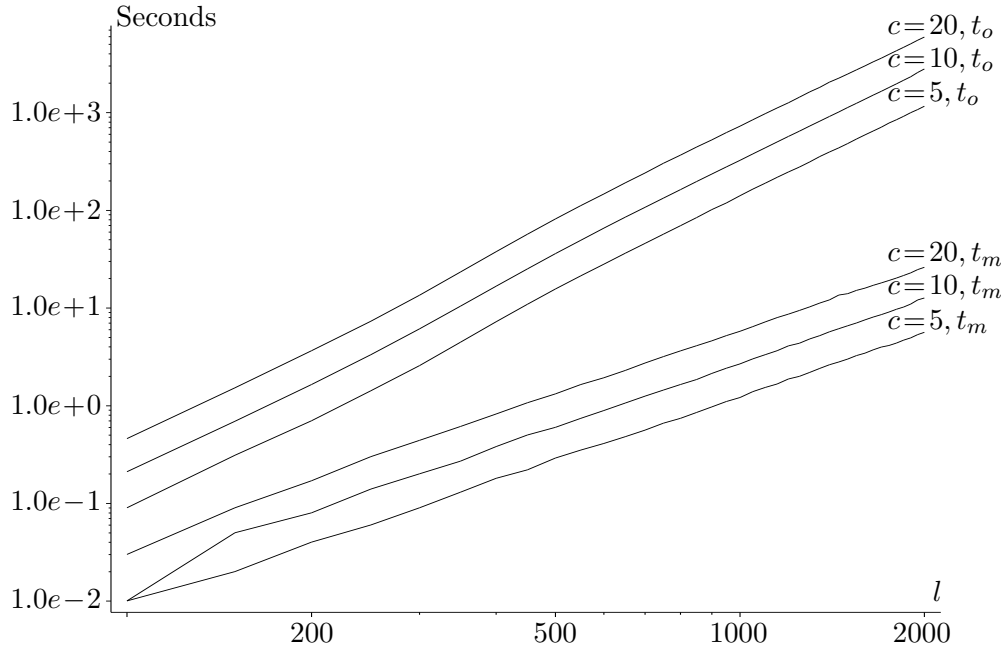


Figure 3.13: CPU times t_o and t_m for $0 < l \leq 2000$ and Linearly Increasing t_i

is used for both axes. The run time t_o of the original algorithm increases very rapidly. For example t_o exceeds 2 minutes for all $l \geq 1000$ and $c = 10, 20$. The run time t_m is never greater than 30 seconds for all $l \leq 2000$ and $c = 5, 10, 20$. One can conclude that the temporal complexity is reduced drastically using the modified algorithm.

In order to get a better impression of the improvement achieved, the ratio t_o/t_m is studied. The behavior of the ratio t_o/t_m is similar for all considered distributions of the step costs. Therefore Fig. 3.14 reports this ratio only for the random distributed step costs and the different number of checkpoints available. As can be seen the ratio shows a linear behavior for $l \leq 2000$. Hence one can conclude that the modified search algorithm has a time complexity of only $\mathcal{O}(cl^2)$ for the considered examples of step costs. The improvement is caused by the fact that the interval to search is drastically reduced in consequence of Theorem 3.3. This is illustrated by Table 3.2 that shows the frequency of

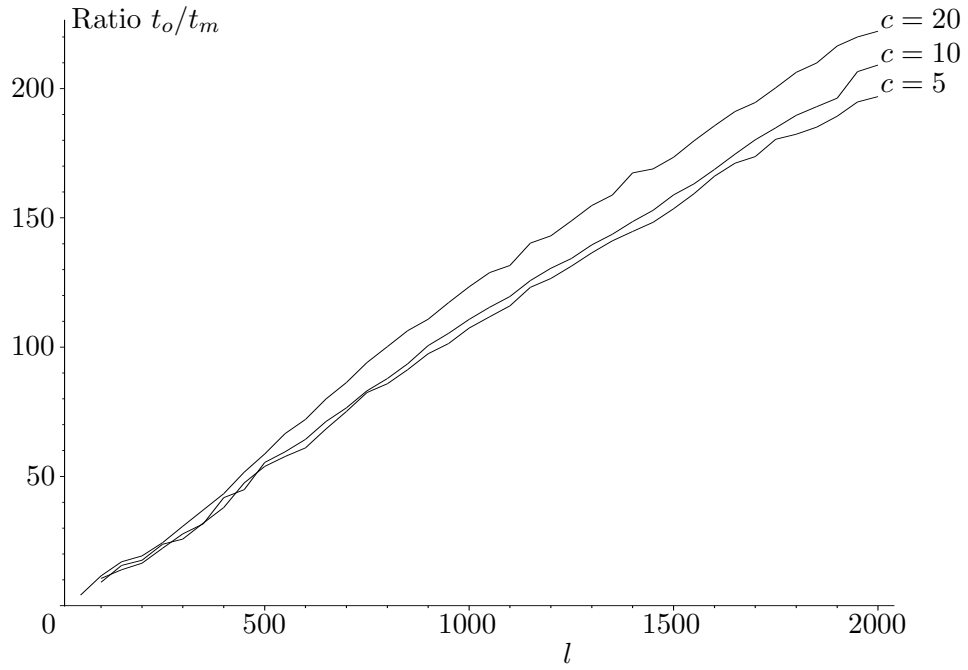


Figure 3.14: Ratio t_o/t_m for $0 < l \leq 2000$ and Random Distributed Step Costs

different interval sizes for $l = 2000$ and $c = 5, 10, 20$ with s denoting the length of the search interval. If the sizes s could be uniformly bounded in general, i.e. for an arbitrary distribution of step costs, one would have the temporal complexity of only $\mathcal{O}(cl^2)$ for the modified search algorithm.

Linear Increase

$s \backslash c$	5	10	20
$= 0$	3.50 %	5.40 %	5.50 %
$= 1$	96.05 %	94.01 %	94.21 %
$= 2$	0.40 %	0.40 %	0.24 %
$= 3$	0.04 %	0.08 %	0.04 %
> 3	0.01 %	0.02 %	0.01 %

Linear Decrease

$s \backslash c$	5	10	20
$= 0$	0.01 %	0.12 %	0.51 %
$= 1$	99.98 %	99.76 %	99.13 %
$= 2$	0.01 %	0.11 %	0.26 %
$= 3$	0.00 %	0.01 %	0.09 %
> 3	0.00 %	0.00 %	0.01 %

Gaussian Distribution

$s \backslash c$	5	10	20
$= 0$	11.99 %	16.72 %	16.10 %
$= 1$	87.89 %	83.17 %	83.82 %
$= 2$	0.10 %	0.10 %	0.07 %
$= 3$	0.02 %	0.01 %	0.01 %
> 3	0.00 %	0.00 %	0.00 %

Random Distribution

$s \backslash c$	5	10	20
$= 0$	91.1 %	92.3 %	91.3 %
$= 1$	1.2 %	1.0 %	1.4 %
$= 2$	0.8 %	0.6 %	0.8 %
$= 3$	0.7 %	0.5 %	0.7 %
> 3	6.2 %	5.6 %	5.8 %

Table 3.2: Search Interval Sizes in Improved Search Algorithm

3.4.3 Conclusions

Having the new theoretical results of this thesis at hand it is possible to reduce the run time of the search algorithm to find an optimal serial reversal schedule

for non-uniform step costs drastically. For example if one has 3000 physical steps with linearly increasing step costs to be reversed and 10 checkpoints that can be used the CPU-time needed for the search is shortened from over one hour to only 42 seconds on a Pentium II PC with 350 Mc/s and 256 MByte RAM.

The run times achieved with the modified algorithm for the quite different step costs considered in the last subsection suggest that a general reduction of the temporal complexity from $\mathcal{O}(cl^3)$ to $\mathcal{O}(cl^2)$ is possible using the improvements derived above. Also the sizes of the search intervals cause this presumption. Nevertheless, so far there is no proof that the temporal complexity of the modified algorithm is $\mathcal{O}(cl^2)$. This will be the subject of further work in the future.

Chapter 4

Parallel Reversal Schedules

4.1 Introduction and Notations

On one hand serial reversal schedules allow an enormous reduction of the memory required to reverse a given evolutionary system F with the basic approach described in the previous chapter. On the other hand one has to pay for this improvement in form of a greater temporal complexity. If any increase of the time needed to reverse F is not acceptable the usage of sufficient processors provides the possibility to reverse the evolutionary system F with drastically reduced spatial complexity and minimal temporal complexity. Therefore this chapter has the development of optimal parallel reversal schedules for a given number of processors and checkpoints as subject.

It will be assumed throughout that the one-step evolution F to be reversed consists of l physical steps with uniform step costs $t_i = \omega \in \mathbb{R}$. Furthermore, it will be supposed that also the time $\hat{t}_i$ needed to perform the recording steps $\hat{F}_i$ and the time $\bar{t}_i$ needed to perform the reverse steps $\bar{F}_i$ are constant, i.e. there exist $\hat{\omega}, \bar{\omega} \in \mathbb{R}$ with $\hat{t}_i = \hat{\omega}$ and $\bar{t}_i = \bar{\omega}$ for all $0 \leq i < l$. Then ω may serve as measuring unit in time and one has that

$$\hat{t}_i = \hat{t}\,\omega \qquad \text{and} \qquad \bar{t}_i = \bar{t}\,\omega$$

with some $\hat{t}, \bar{t} \in \mathbb{R}$. Because each recording step $\hat{F}_i$ stores the intermediates needed for the reverse step onto the tape besides the evaluation of the physical step F_i it follows that $\hat{t} \geq 1$. Furthermore, it is assumed that the evaluation of one reverse step $\bar{F}_i$ is at least as time consuming as the evaluation of one physical step F_i and hence $\bar{t} \geq 1$. Using these assumptions without loss of generality one may normalize the temporal complexities to

$$t_i = 1, \qquad \hat{t}_i = \hat{t} \geq 1, \qquad \text{and} \qquad \bar{t}_i = \bar{t} \geq 1$$

for all $0 \leq i < l$. Here, one faces the first big difference between serial and parallel reversal schedules. Only one processor is available to perform the recording steps and the reverse steps in the serial case. Hence it does not matter whether they have the same temporal complexity or not because nothing else can be done while a recording step or a reverse step is performed. If more than one processor is used in the reversal it is important to know the time required to

perform a particular recording or reverse step in order to manage the other processors appropriately.

Suppose the given evolutionary system F comprises l physical steps and determines the temporal complexities $\hat{t}$ and $\bar{t}$. Then the minimal time t_M required by a parallel reversal schedule S to reverse F equals

$$t_M \equiv (l-1) + \hat{t} + l\,\bar{t} = (\bar{t}+1)l + \hat{t} - 1$$

because one has to perform at least one forward sweep from the initial state 0 to the penultimate state $l-1$. Since $t_i \equiv 1$, at least $l-1$ time units are needed for that. Then the recording step $\hat{F}_{l-1}$ has to be performed. This requires $\hat{t}$ time units. Because each reverse step $\bar{F}_i$ is based on the results of the reverse step $\bar{F}_{i+1}$ for $l-1 > i \geq 0$ they cannot be parallelized. Therefore the minimal time to perform the l reverse steps that are needed is given by $l\,\bar{t}$. Hence it is shown that t_M equals $(l-1) + \hat{t} + l\,\bar{t}$. This minimal temporal complexity can be used to characterize parallel reversal schedules.

Definition 4.1 (Feasible Parallel Reversal Schedule). Let F be the one-step evolution under consideration that combines l physical steps and determines the parameters $\hat{t}$ and $\bar{t}$. A parallel reversal schedule S to reverse F is called *feasible* if the time needed by S to perform the reversal of F equals $t_M = (l-1) + \hat{t} + l\,\bar{t}$.

Hence a feasible parallel reversal schedule is optimal with respect to the run time needed for reversing F, but not necessarily with respect to the processors and checkpoints required.

From the formula $(\bar{t}+1)l + \hat{t} - 1$ follows that the value $\bar{t}$ has an important influence on the minimal time t_M for reversing a given sequence of physical steps. Therefore it make sense to reduce $\bar{t}$ at the expense of $\hat{t}$. For example one can use the technique of preaccumulation (see e.g. [Gri00] for adjoint computation). Here, all intermediates that do not depend on the value of the previous reverse step but are needed for the next one are calculated already by the corresponding recording step. In case of adjoint calculations the preaccumulation could evaluate for instance local derivatives. Thus it becomes possible to reduce $\bar{t}$ to 1 by modifying the recording steps appropriately. Hence it is important to examine the case $\bar{t} = 1$. Moreover, on this account $\hat{t} \gg 1$ represent a relevant research topic, because $\hat{t}$ could possibly be increased drastically by preaccumulation.

For developing feasible parallel reversal schedules an obvious idea could be to use a bisection strategy.

Example 4.1 *(Bisection strategy).* If the one-step evolution F under consideration combines 8 physical steps with $\hat{t} = \bar{t} = 1$ this technique yields the feasible parallel reversal schedule shown in Fig. 4.1. Here, 4 processors and 3 checkpoints are employed to perform the reversal of F. The initial state is copied to the first checkpoint. Then a processor performs a forward sweep to the 4th state, i.e. the midpoint of 9 states. This state is copied to the second checkpoint. After that a forward sweep is performed to the 6th state, i.e. the midpoint of state 4 and 8. The 6th state is copied to the third checkpoint. After another physical step the first recording step is performed. Subsequently the calculation

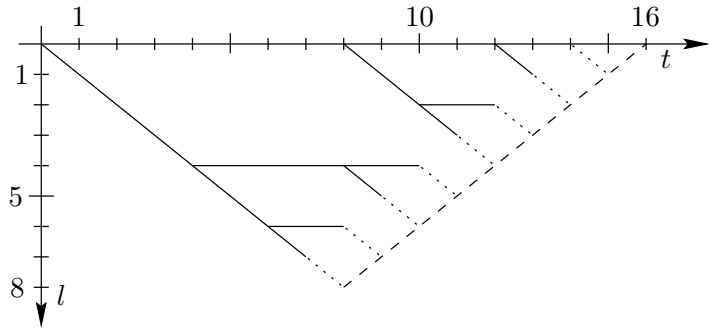


Figure 4.1: Parallel Reversal Schedule using Bisection Strategy

of the reversal of F starts. The number of processors needed to execute this reversal schedule is given by the maximal number of slanted lines crossing any vertical line. Therefore three processors are required by the parallel reversal schedule displayed in the last figure.

The only published article dealing with parallel reversal schedules so far considers a slightly more general idea, namely an uniform distribution of the checkpoint positions [Ben96]. The corresponding implementation DAP [Ben95] written in C uses PVM [GBD+94] to control the processors and to manage the communication between them.

Naturally, the question arises if the parallel reversal schedules determined by the bisection strategy or the recursive partition with fixed ratio as in [Ben96] are optimal in terms of the number of processors and the number of checkpoints required. The answer is "No" because, for example, 3 processors and 2 checkpoints suffice to reverse the evolutionary system F of Example 4.1 as shown in Fig. 4.2. Here, a slightly different reversal technique was chosen, namely a

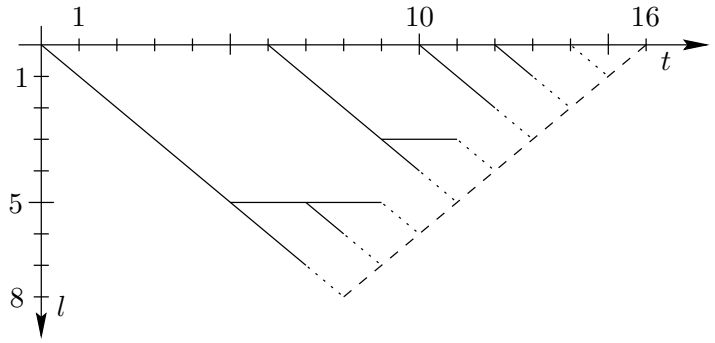


Figure 4.2: Improved Parallel Reversal Schedule

checkpoint writing of the fifth and subsequently of the third state.

Obviously, one would like to find an optimal reversal schedule, i.e. one that requires a minimal number of resources. As a first step this chapter determines the maximal number of physical steps that can be reversed with a given number of processors and checkpoints. For the development of the corresponding

optimal reversal schedules from now on only uniform one-step evolutions are considered. Hence the temporal complexity of all recording steps and all reverse steps are given by the integer constants $\hat{t}, \bar{t} \in \mathbb{N}\backslash\{0\}$, respectively (see Def. 2.5). Naturally, this means a restriction of the evolutionary systems under consideration, but allows the following

Definition 4.2 (Computational Cycle). The time needed to reverse l physical steps with the corresponding temporal complexities $\hat{t}$, $\bar{t} \in \mathbb{N}$ using a feasible parallel reversal schedule can be divided into $l-1+\hat{t}+l\,\bar{t}$ *computational cycles*. During each computational cycle at least one complete physical step F_i, one part of a recording step $\hat{F}_i$, or one part of a reverse step $\bar{F}_i$ is performed.

In order to simplify the notation in the proofs of this chapter the computational cycles are numbered in reverse order. This will be illustrated by the next example.

Example 4.2 *(Reversal Schedule for $l = 9$, $\hat{t} = 2$, and $\bar{t} = 3$).* Suppose the uniform one-step evolution F under consideration consists of 9 physical steps. Furthermore, one has $\hat{t} = 2$ and $\bar{t} = 3$ as temporal complexities for the recording and reverse steps, respectively. Then Fig. 4.3 shows the reversal trace belonging to one feasible parallel reversal schedule to reverse F with 3 processors and 2 checkpoints. Obviously, there are $37 = 8+2+9\cdot 3$ computational cycles j with

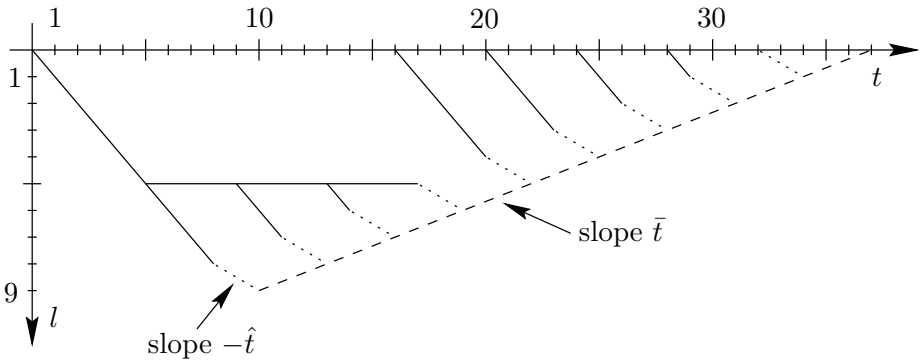


Figure 4.3: Feasible Parallel Reversal Schedule for $l = 9$, $\hat{t} = 2$, and $\bar{t} = 3$

$j = -t+38$. Because of this reverse numbering in the first computational cycle $j = 1$ only one processor performs the last part of the reverse step $\bar{F}_0$, in the 9th computational cycle one checkpoint stores the initial state 0, one processor performs F_0, and another one a part of the reverse step $\bar{F}_2$. During the last, i.e. the 37th, computational cycle one checkpoint stores the initial state and one processor performs F_0. Hence the number of the computational cycles are in reverse order compared to the numbering of the time units needed for the reversal. As mentioned before this has only technical reasons in the notation of the proofs to follow.

In order to analyze the parallel reversal schedules in more detail it will be studied how many processors and checkpoints are used in each computational cycle. For that purpose one can use a

Definition 4.3 (Resource Profile $R(S)$). Suppose S is a feasible parallel reversal schedule for the uniform one-step evolution F. Let F comprise l physical steps and determine the temporal complexities $\hat{t}$ and $\bar{t}$. The *resource profile* $R(S)$ with

$$R(S) \quad \equiv \quad \left(c^j,\, p^j,\, s^j\right)_{1 \le j \le (l-1)+\hat{t}+l\,\bar{t}}$$

specifies for the computational cycle j, $1 \le j \le (l-1) + \hat{t} + l\,\bar{t}$, the number c^j of checkpoints used in j, the number p^j of processors used in j, and the sum $s^j = c^j + p^j$ of both.

Example 4.3 *(Possible Resource Profile for $l = 5$, $\hat{t} = 1$, and $\bar{t} = 1$).* The given uniform one-step evolution F comprises 5 physical steps and determines the temporal complexities $\hat{t} = 1$ and $\bar{t} = 1$. Figure 4.4 shows one possible parallel reversal schedule S for F. Furthermore, the corresponding resource

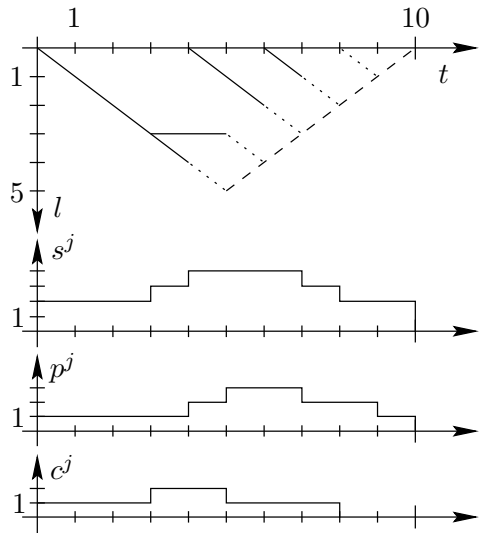


Figure 4.4: Resource Profile

profile $R(S)$ is illustrated. As can be seen first s^j increases monotonously in a warm-up phase, then reaches a plateau, and finally decreases monotonically in a cool-down phase. A corresponding behavior can be observed for parallel reversal schedules that reverse the maximal number of physical steps for a given number of processors and checkpoints that are available.

Definition 4.4 (Ordering of Resource Profiles). Suppose the given uniform one-step evolution F comprises l physical steps and determines the temporal complexities $\hat{t}$ and $\bar{t}$. Let S_1 and S_2 be two parallel reversal schedules for F with the same number of computational cycles. The parallel reversal schedule S_1 uses no more resources than S_2 if one has

$$R(S_1) \ni s^j \quad \le \quad s^j \in R(S_2) \quad \text{ for all } \quad j \in \{1, \ldots, (l-1) + \hat{t} + l\,\bar{t}\}\ .$$

This will be denoted as $R(S_1) \le R(S_2)$.

Throughout this chapter it is assumed that the available resources have the following property:

Definition 4.5 (Processor-Checkpoint Convertibility). If a resource can be used in each computational cycle either as checkpoint or as processor it has the property of *processor-checkpoint convertibility*.

The remainder of this chapter is organized as follows. In Section 4.2 some basic properties of feasible parallel reversal schedules are shown. It is assumed that during each computational cycle k resources each of which can be used either as processor or as checkpoint are available. For all $k \in \mathbb{N}$ and given temporal complexities $\hat{t}$ and $\bar{t}$ an upper bound of the maximal number l_k of physical steps that can be reversed with k resources is proven.

In Section 4.3 feasible parallel reversal schedules are constructed that attain the proven upper bound of physical steps for k resources with the property of processor-checkpoint convertibility. This is a rather technical part of this thesis. Then the maximal number of physical steps that can be reversed with a given number of resources is determined.

Finally, Section 4.4 presents some illustrations of the results achieved and draws several conclusions.

In order to get a first idea of the maximal number l_k of physical steps that can be reversed with up to k resources an exhaustive search algorithm was applied. This algorithm constructed implicitly all possible parallel reversal schedules for a given number of processors and checkpoints to determine the maximal number of physical steps that could be reversed with these resources. In spite of the fact that a sophisticated storage method was used to decide if a particular parallel reversal schedule has been examined already the time needed to determine the maximal number of physical steps that can be reversed grew exponentially with the number of processors and checkpoints available. Even on a SGI Origin 2000 it was not possible to find for $\hat{t} = \bar{t} = 1$ the maximal number l_k of physical steps that can be reversed for more than 9 resources. Nevertheless the available results for $k \leq 9$ resources that have the property of processor-checkpoint convertibility provided sufficient insight to develop the theory presented in the Sections 4.2 and 4.3.

4.2 Structural Properties and an Upper Bound

For the serial reversal schedules considered in the previous chapter the property of checkpoint persistence yielded the possibility to split a serial reversal schedule into two parts. This splitting forms an important ingredient for the proofs of the minimal number of physical steps and the minimal number of checkpoint writings performed during the reversal. In order to analyze feasible parallel reversal schedules the same strategy is applied, namely the splitting of one feasible parallel reversal schedule into smaller ones. Therefore the following definition describes how two feasible parallel reversal schedules can form another one.

Definition 4.6 (Composition of Parallel Reversal Schedules). Assume the physical steps under consideration determine the temporal complexities $\hat{t} \in \mathbb{N}$ to perform one recording step and $\bar{t} \in \mathbb{N}$ to perform one reverse step. Let S_1 and S_2 be two feasible parallel reversal schedules to reverse l_1 and l_2 of the given physical steps, respectively. The composition $S = S_1 + S_2$ of S_1 and S_2 is defined as a feasible parallel reversal schedule S that reverses $l \equiv l_1 + l_2$ physical steps in $(\bar{t}+1)l + \hat{t} - 1$ computational cycles by performing the following tasks:

i: Copy state 0 to a checkpoint in the computational cycle $(\bar{t}+1)l + \hat{t} - 1$.

ii: Perform a forward sweep from the initial state 0 to state l_2 during the computational cycles $(\bar{t}+1)l + \hat{t} - 1, \ldots, (\bar{t}+1)l + \hat{t} - l_2$.

iii: Start S_1 at state l_2 to reverse the physical steps $l_2, \ldots, l-1$ in the computational cycle $j_1 \equiv (\bar{t}+1)l + \hat{t} - 1 - l_2$.

iv: Start S_2 at the initial state 0 to reverse the physical steps $0, \ldots, l_2 - 1$ in the computational cycle $j_2 \equiv (\bar{t}+1)l_2 + \hat{t} - 1$.

Figure 4.5 depicts the resulting parallel reversal schedule S for given S_1 and S_2 each of which is illustrated with dash-doted lines. As can be seen S is

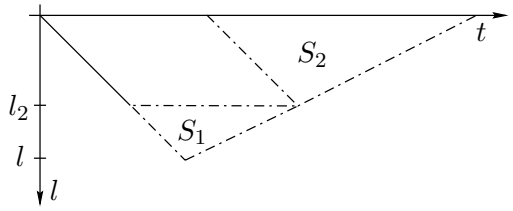


Figure 4.5: Parallel Reversal Schedule $\tilde{S}$

feasible. The computational cycle j_1 and j_2 chosen in tasks iii and iv to start S_1 and S_2, respectively, ensures that the reverse steps can be performed without an interruption. Moreover one finds that the composition of feasible parallel reversal schedules is neither associative nor commutative.

The resource profile $R(S)$ of $S = S_1 + S_2$ can be determined easily. In detail one obtains with $(c^{j,i},\, p^{j,i},\, s^{j,i}) \in R(S_i)$ for the computational cycles $1 \leq j \leq \bar{t}\, l_2$ of S that

$$R(S) \ni \left(c^j,\, p^j,\, s^j\right) = \left(c^{j,2},\, p^{j,2},\, s^{j,2}\right) \ .$$

For $\bar{t}\, l_2 < j \leq \min\{j_1, j_2\}$, i.e. the overlapping region of S_1 and S_2, follows

$$R(S) \ni \left(c^j,\, p^j,\, s^j\right) = \left(c^{j-\bar{t}\, l_2,1} + c^{j,2},\, p^{j-\bar{t}\, l_2,1} + p^{j,2},\, s^{j-\bar{t}\, l_2,1} + s^{j,2}\right) \ .$$

If $j_2 < j_1$ the parallel reversal schedule S_1 starts temporally before S_2. Hence one has for $\min\{j_1, j_2\} < j \leq \max\{j_1, j_2\}$

$$R(S) \ni \left(c^j,\, p^j,\, s^j\right) = \left(c^{j-\bar{t}\, l_2,1},\, p^{j-\bar{t}\, l_2,1},\, s^{j-\bar{t}\, l_2,1}\right) \ .$$

Otherwise it follows for $\min\{j_1, j_2\} < j \leq \max\{j_1, j_2\}$ that

$$R(S) \ni \left(c^j,\, p^j,\, s^j\right) = \left(c^{j,2},\, p^{j,2},\, s^{j,2}\right) \ .$$

During the remaining computational cycles only one processor and one checkpoint are needed. Hence one finds for $\max\{j_1, j_2\} < j \leq (\bar{t}+1)l + \hat{t} - 1$ that

$$R(S) \ni \left(c^j,\, p^j,\, s^j\right) = (1,\, 1,\, 2) \ .$$

For serial reversal schedules one has the property of checkpoint persistence as mentioned above. Throughout this chapter is assumed that there are k available resources, which have the property of processor-checkpoint convertibility. Then a quite similar property, therefore also called "Checkpoint Persistence", can be proven for parallel reversal schedule.

Lemma 4.1 (Checkpoint Persistence).
Let $\tilde{S}$ be a parallel reversal schedule to reverse l physical steps. Assume that while executing $\tilde{S}$ a checkpoint converts into a processor in order to perform a forward sweep and is used subsequently again as checkpoint. Then there exists

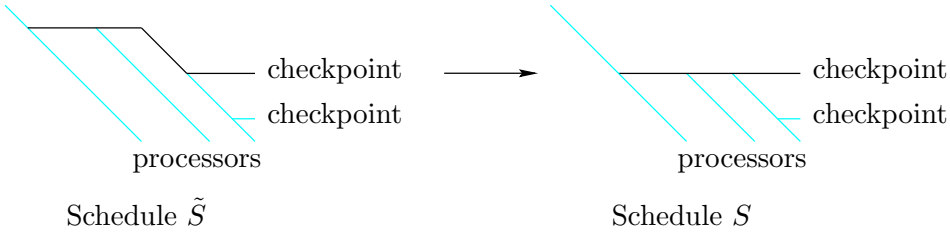


Figure 4.6: Transformation of $\tilde{S}$ into S

a parallel reversal schedule S with $R(S) \leq R(\tilde{S})$ to reverse l physical steps such that the checkpoint is used only as checkpoint.

Proof. Figure 4.6 depicts this situation, where the checkpoint is drawn as black line. The behavior of the particular checkpoint in the parallel reversal schedule $\tilde{S}$ is illustrated again by Fig. 4.7. In the grey region each forward sweep starting

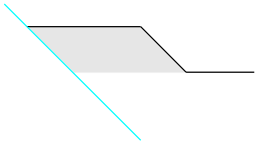


Figure 4.7: Domain of $\tilde{S}$ to Change

there and reaching the lower boundary of the grey region can start also correspondingly from the modified checkpoint of S. Hence all other forward sweeps and each checkpoint writing in the grey domain are then useless and can be deleted. Obviously the number of computational cycles needed is the same for $\tilde{S}$ and S.

For the computational cycles that are different in $\tilde{S}$ and S one finds the following facts. The parallel reversal schedule S needs one processor to evaluate the part of the forward sweep that forms the left boundary of the grey region and subsequently one checkpoint that represents the lower boundary of the grey domain. In the corresponding computational cycles $\tilde{S}$ utilizes at least one checkpoint that forms the upper boundary of the grey domain and after that at least one processor in order to perform the forward sweep that represents the right boundary of the grey domain. Therefore during the modified computational cycles the number of resources used by S is less than or equal to the number of resources used by $\tilde{S}$. ∎

With respect to the processors utilized in a parallel reversal schedule one can show the following corresponding property:

Lemma 4.2 (Processor Persistence).
Let $\tilde{S}$ be a parallel reversal schedule to reverse l physical steps. Assume that while executing $\tilde{S}$ a processor performs a forward sweep, converts into a checkpoint, and is used subsequently again for a forward sweep. Then there exists a

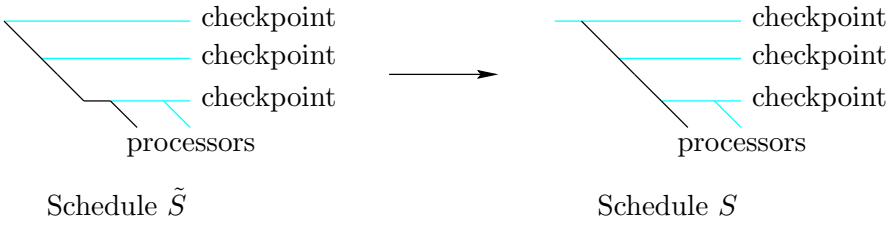


Figure 4.8: Transformation of $\tilde{S}$ into S

parallel reversal schedule S with $R(S) \leq R(\tilde{S})$ to reverse l physical steps such that the processor is used only for the two forward sweeps.

Proof. Figure 4.8 depicts this situation, where the processor is drawn as black line. The behavior of the particular processor in the parallel reversal schedule $\tilde{S}$ is illustrated again by Fig. 4.9. In the grey region each checkpoint writing that

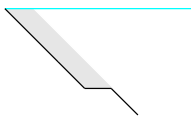


Figure 4.9: Domain of $\tilde{S}$ to Modify

is performed can be executed also during the forward sweep in S that forms the right boundary of the grey region if the stored state is used in the further reversal process. Therefore all other checkpoint writings and each forward sweep in the grey domain are useless and can be deleted. Obviously the number of computational cycles needed is the same for $\tilde{S}$ and S.

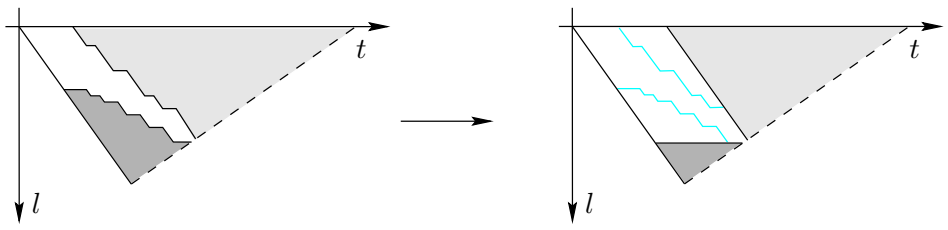
Consider the computational cycles that are different in $\tilde{S}$ and S. During the execution of these computational cycles S uses one checkpoint that forms the

upper boundary of the grey domain and subsequently one processor in order to perform the forward sweep that represents the right boundary of the grey domain. For the evaluation of the same computational cycles $\tilde{S}$ uses at least one processor performing the forward sweep that forms the left boundary of the grey domain. After that at least one checkpoint that represents the lower boundary of the grey domain is used by $\tilde{S}$. Hence during the modified computational cycles the number of resources used by S is less than or equal to the number of resources used by $\tilde{S}$. ∎

From the last two lemmas one can conclude without loss of generality that in a parallel reversal schedule a forward sweep ends always at an appropriate recording step. The same is true for each checkpoint writing, i.e. a checkpoint is held until the state saved in this checkpoint is used to start a recording step. Moreover, the last two lemmas are applied to prove the following important property of feasible parallel reversal schedules, namely the splitting into two smaller ones.

Theorem 4.1 (Binary Decomposition).
For arbitrary temporal complexities $\hat{t}, \bar{t} \in \mathbb{N}$ let $\tilde{S}$ be a feasible parallel reversal schedule to reverse $l \geq 2$ physical steps. Then there exists a feasible parallel reversal schedule S for the reversal of l physical steps, which is the composition of two feasible parallel reversal schedules S^1 and S^2. Furthermore, $R(S) \leq R(\tilde{S})$ is valid.

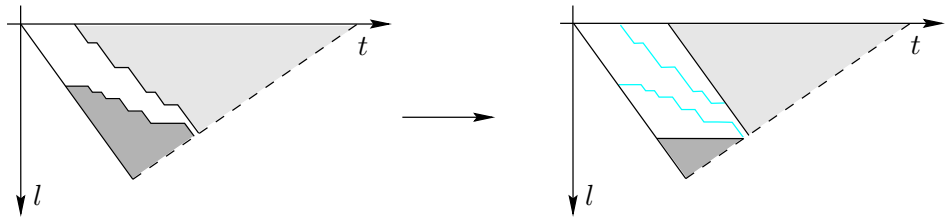
Proof. If $\tilde{S}$ satisfies already the assertion nothing needs to be shown. Therefore assume that $\tilde{S}$ does not have the desired structure. Then $\tilde{S}$ can be easily transformed to one of the feasible parallel reversal schedules $\tilde{S}$ depicted in Fig. 4.10 or in Fig. 4.11. Here, at least one checkpoint writing is performed by the forward sweep from the initial state to state $l - 1$. It is not of interest



Schedule $\tilde{S}$ Schedule S

Figure 4.10: First Possible Structure of $\tilde{S}$ and the Resulting S

what happens inside the dark grey and light grey domain of $\tilde{S}$, respectively, but it is easy to simplify them to the dark grey and light grey domain of S. The properties checkpoint persistence and processor persistence shown in the last two theorems can be applied. Therefore, the parallel reversal schedule S can be constructed as depicted in Fig. 4.10 and Fig. 4.11, respectively, according to the proofs of the Theorems 4.1 and 4.2. This yields the desired structure of S, because then S is given as composition of two feasible parallel reversal



Schedule $\tilde{S}$ Schedule S

Figure 4.11: Second Possible Structure of $\tilde{S}$ and the Resulting S

schedules S^1 and S^2. Moreover, it is proven that each computational cycle of S needs no more processors and checkpoints that the corresponding one of $\tilde{S}$. ∎

As a second important property of parallel reversal schedules it is possible to show that the number of processors and checkpoints grows monotonously from the first checkpoint writing up to the first recording step. This assertion will be proven in the next

Theorem 4.2 (Monotony of Resource Requirements before Vertex). *For arbitrary temporal complexities $\hat{t}, \bar{t} \in \mathbb{N}$ let $\tilde{S}$ be a feasible parallel reversal schedule for l physical steps. Then there exists a feasible parallel reversal schedule S to reverse l physical steps, such that $R(S) \leq R(\tilde{S})$. Moreover each computational cycle $j = \bar{t}\,l + 2, \ldots, (\bar{t}+1)l + \hat{t} - 1$ of S needs no more resources than the preceding computational cycle $j-1$ of S. I.e. the resource profile $R(S)$ is monotonously decreasing in the last $l + \hat{t} - 1$ computational cycles.*

Proof. If $\tilde{S}$ has already the monotonously decreasing resource profile nothing has to be proven.

Therefore assume that the assertion does not hold for $\tilde{S}$. An appropriate parallel reversal schedule S can be constructed using the given schedule $\tilde{S}$ in the following way. First define $S \equiv \tilde{S}$. Determine the minimal computational cycle $j_l \in \{\bar{t}\,l + 2, \ldots, (\bar{t}+1)l + \hat{t} - 1\}$ of $\tilde{S}$, where more processors and checkpoints are needed than in the computational cycle $j_l - 1$. This j_l exists because $\tilde{S}$ does not fulfil the assertion.

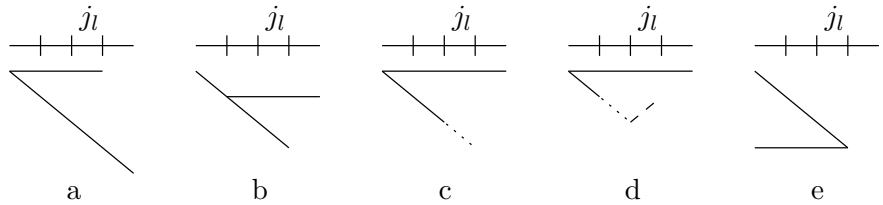


Figure 4.12: Reasons for Increase in Applied Resources

Figure 4.12 shows all possibilities that could cause an increase in the number of resources that are used. First one state could be stored in a checkpoint during

j_l but this state is not used in $j_l - 1$ (see possibility a). One can delete this checkpoints in j_l of S without any effect on the reversal process. Second a processor could perform a physical step, a recording step or a reverse step the result of which is not used in $j_l - 1$ (see possibilities b, c, d). Therefore one can delete this evaluation in j_l of S without any effect on the reversal process. Third a processor and a checkpoint could merge in j_l (see possibility e). Then either the processor or the checkpoint can be deleted in j_l of S without any influence on the reversal process.

These modifications can be repeated no more than $l + \hat{t} - 3$ times. As a consequence of the elimination process one has that each computational cycle j with $j = \bar{t}\,l + 2, \ldots, (\bar{t}+1)l + \hat{t} - 1$ of the feasible parallel reversal schedule S needs no more processors and checkpoints than the preceding computational cycle $j-1$. Furthermore, it is obvious that a computational cycle of S does not use more processors and checkpoints than the corresponding one of $\tilde{S}$. Hence the assertion is proven. ∎

Having these theoretical results at hand, it is possible to derive the main result of the section. An upper bound for the maximal number of physical steps that can be reversed with up to k resources available in each computational cycle will be proven. As throughout it is supposed that during a computational cycle a resource can be utilized either as processor or as checkpoint.

Theorem 4.3 (Upper Bound of Physical Steps to Be Reversed).
For arbitrary temporal complexities $\hat{t}, \bar{t} \in \mathbb{N}$ let $k \in \mathbb{N}$ denote the number of available resources each of which has the property of processor-checkpoint convertibility. For the maximal number l_k of physical steps that can be reversed by any conceivable feasible parallel reversal schedule with up to k resources in each computational cycle the following upper bound is valid:

$$l_k \;\leq\; \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\,l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} . \tag{4.1}$$

Proof. First it will be shown that $l_k \leq k$ for $k < 2 + \hat{t}/\bar{t}$. Second an induction on k is used to prove $l_k \leq l_{k-1} + \bar{t}\,l_{k-2} - \hat{t} + 1$ for $k \geq 2 + \hat{t}/\bar{t}$. Thus the assertion will be verified.

Let k be smaller than $2 + \hat{t}/\bar{t}$, i.e. $k \leq 2 + \lfloor(\hat{t} - 1)/\bar{t}\rfloor$. First assume that $\hat{t} \leq \bar{t}$ and therefore $k \in \{1, 2\}$. In this case it is obvious that for $k = 1$ only one physical step can be reversed and for $k = 2$ only two physical steps. Now suppose that $\hat{t} > \bar{t}$. Let S be a feasible parallel schedule to reverse l_k physical steps with up to k resources. Set $j \equiv (\bar{t} - 1)l_k + 1$, i.e. j denotes the computational cycle during which the last part of the reverse step $\bar{F}_{l_k - 1}$ is performed. One processor has to perform this part of $\bar{F}_{l_k - 1}$. To make the results for the next reverse steps available in time $r \equiv \min\{k - 1, \lceil \hat{t}/\bar{t}\,\rceil\}$ processors are applied to perform the corresponding recording steps in the computational cycle j. Hence if $r = k - 1$ it follows that $r + 1 = k$ processors are needed for the recording steps and the reverse step. Therefore no checkpoint can store the initial state in the computational cycle j. For that reason it is impossible to start a forward sweep or a recording step from the initial state anymore. Hence

only k physical steps can be reversed. If $r = \lceil \hat{t}/\bar{t} \rceil = \lfloor (\hat{t}-1)/\bar{t} \rfloor + 1$ one has $\lfloor (\hat{t}-1)/\bar{t} \rfloor + 1 \leq k-1 \leq 1 + \lfloor (\hat{t}-1)/\bar{t} \rfloor$. Hence $\lfloor (\hat{t}-1)/\bar{t} \rfloor + 2 = k$ processors are needed to perform the recording steps and the reverse step. Once more no checkpoint can store the initial state in the computational cycle j and it is impossible to start a forward sweep or a recording step from the initial state anymore. Therefore again only k physical steps can be reversed.

Now let k be equal to $2 + \lceil \hat{t}/\bar{t} \rceil$. It must be shown that the inequality $l_k \leq (\bar{t}+1)k - 2\bar{t} - \hat{t}$ is valid. Suppose S is a feasible parallel reversal schedule that reverses l_k physical steps with up to k resources. Consider again the computational cycle j with $j \equiv (\bar{t}-1)l_k + 1$, where the last part of the reverse step $\bar{F}_{l_k-1}$ is performed. In the same computational cycle $r \equiv \lceil \hat{t}/\bar{t} \rceil$ processors are applied to perform the recording steps needed to provide the data for the next reverse steps in time. Hence only $k-1$ processors are used for the recording steps and the reverse step. Therefore a checkpoint can store the initial state. In the computational cycle $j \equiv (\bar{t}-1)l_k$ the processor performing $\hat{F}_{l_k-2}$ becomes available. Because all other processors perform a recording or a reverse step, respectively, the free processor cannot be used as a checkpoint elsewhere. Hence the only possibility is to use this free processor to perform a forward sweep starting at the initial state succeeded by a recording step. This forward sweep and the recording step require $l_k - k + \hat{t}$ computational cycles. It is finished in time such that the reversal can be executed without any interruption if $l_k - k + \hat{t} = (k-2)\bar{t}$ because $(k-2)\bar{t}$ computational cycles are needed for the $k-2$ reverse steps evaluated after $\hat{F}_{l_k-2}$ is completed. The last equality yields $l_k = (\bar{t}+1)k - 2\bar{t} - \hat{t}$ and hence $l_k \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1$ is proven.

If $k > 2 + \lceil \hat{t}/\bar{t} \rceil$ an induction on k will be used to show the upper bound (4.1). Since $k > 3$ it follows that $l_k > 3$. Hence, among all feasible parallel reversal schedules to reverse the maximal number l_k of physical steps with up to k processors and checkpoints applied at any time there is at least one feasible parallel reversal schedule S that is a composition of two feasible parallel reversal schedules S^1 and S^2 as shown by Fig. 4.13 because of Theorem 4.1. Let l_{S^1}

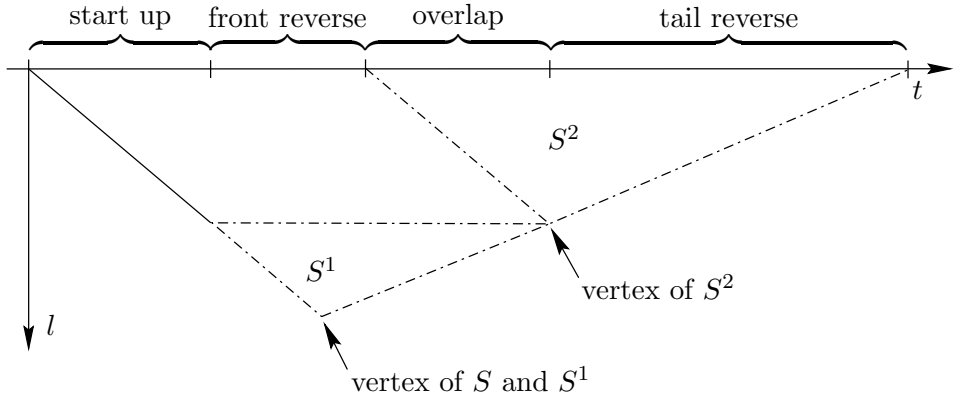


Figure 4.13: Special Structure of S

denote the number of physical steps reversed by S^1 and l_{S^2} the number of physical steps reversed by S^2. One has $l_k = l_{S^2} + l_{S^1}$ as well as $l_{S^1} \leq l_{k-1}$

because one checkpoint stores the initial state during the reversal performed by S_1. Hence at most $k-1$ processors and checkpoints are available in each computational cycle of S^1.

In the remainder of this proof one has to distinguish two possibilities, namely the case $l_{S^1} > l_{k-2}$ and the case $l_{S^1} \leq l_{k-2}$. In the former instance it will be shown that $l_{S^2} \leq \bar{t}\, l_{k-2} - \hat{t} + 1$. In the latter instance the assertion is true if one has $l_{S^2} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - l_{S^1}$. Otherwise a reversal schedule is constructed that fulfils either $l_{S^1} > l_{k-2}$ and $l_{S^2} \leq \bar{t}\, l_{k-2} - \hat{t} + 1$ or $l_{S^1} \leq l_{k-2}$ and $l_{S^2} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - l_{S^1}$.

First suppose that $l_{S^1} > l_{k-2}$ is valid. It follows that there are at least $\bar{t}(l_{k-2}+1)$ computational cycles between the vertex of S^2 and the vertex of S. The computational cycles j of S with $j = \bar{t}\, l_{S^2}+1, \ldots, (\bar{t}+1)l_{S^2}+\hat{t}-1$ form the *overlap* of S^1 and S^2 as depicted in Fig. 4.13. During each computational cycle j in the overlap at least one checkpoint stores the initial state and one processor performs the forward sweep to state $l_{S^2}-1$. It follows that S^1 utilizes in the overlap at most $k-2$ resources. Therefore the first $l_{S^2}+\hat{t}-1$ computational cycles of S^1 employ no more than $k-2$ processors and checkpoints. It will be shown next that $l_{S^2} \leq \bar{t}\, l_{k-2} - \hat{t} + 1$ is valid. Suppose $l_{S^2} > \bar{t}\, l_{k-2} - \hat{t} + 1$, an assumption, which will lead to a contradiction. One obtains

$$l_{S^2} + \hat{t} - 1 > \bar{t} l_{k-2} - \hat{t} + 1 + \hat{t} - 1 = \bar{t} l_{k-2},$$

i.e. $l_{S^2} + \hat{t} - 1 = \bar{t}\, l_{k-2} + r$ with $r \in \mathbb{N}$ and $r \geq 1$. Let $\jmath$ be the counter of the computational cycles in S^1. From the last inequality follows that S^1 needs no more than $k-2$ resources during the computational cycles $\jmath = 1, \ldots, \bar{t}\, l_{k-2}+1$. During these computational cycles the vertex of S^1 is not reached because there are at least $\bar{t}(l_{k-2}+1)$ computational cycles between the vertex of S^1 and the vertex of S^2. These facts are exploited for the construction of a parallel schedule $\tilde{S}^1$ to reverse $l_{k-2}+1$ physical steps such that $\tilde{S}^1$ needs no more than $k-2$ processors and checkpoints at any time. This will yield a contradiction to the definition of l_{k-2}.

The first $\bar{t}\, l_{k-2}+1$ computational cycles of $\tilde{S}^1$ are identical to the first $\bar{t}\, l_{k-2}+1$ computational cycles of S^1. Hence during these computational cycles no more than $k-2$ processors and checkpoints are applied. Let $\tilde{\jmath}$ be the counter of the computational cycles in $\tilde{S}^1$. In the computational cycle $\tilde{\jmath} = \bar{t}\, l_{k-2}+1$ of $\tilde{S}^1$ one processor performs the last part of the reverse step $\bar{F}_{l_{k-2}}$. This processor is used during the $\bar{t}$ computational cycles $\tilde{\jmath} = \bar{t}\, l_{k-2}+1, \ldots \bar{t}\, l_{k-2}+\bar{t}$ for evaluating the reverse step $\bar{F}_{l_{k-2}}$. Furthermore, there are some processors performing recording steps. After the completion of the recording step each processor becomes a checkpoint for the remaining computational cycles. All other processors of the computational cycle $\tilde{\jmath} = \bar{t}\, l_{k-2}+1$ become also checkpoints during the remaining computational cycles of $\tilde{S}^1$. Furthermore, all checkpoints remain checkpoints. The processor performing the reverse step $\bar{F}_{l_{k-2}}$ is used also for the recording step $\hat{F}_{l_{k-2}}$ in the $\hat{t}$ computational cycles $\tilde{\jmath} = \bar{t}\, l_{k-2} + \bar{t} + 1, \ldots, \bar{t}\, l_{k-2} + \bar{t} + \hat{t}$. Subsequently it is utilized for the forward sweep from the initial state to state l_{k-2} during the last l_{k-2} computational cycles of $\tilde{S}^1$. This forward sweep performs all necessary checkpoint writings. The

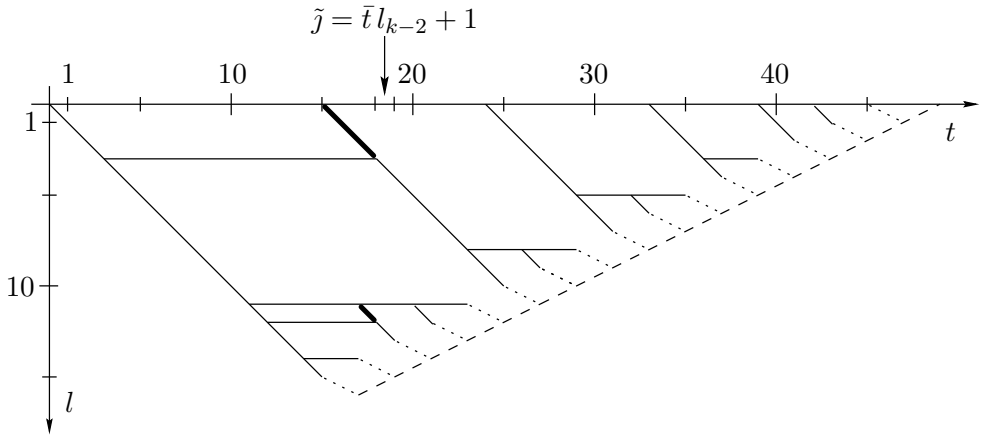


Figure 4.14: Resulting Parallel Reversal Schedule $\tilde{S}^1$

described construction principle is depicted in Fig. 4.14, where the fat slanted lines mark the physical steps performed in S^1. It follows that the feasible parallel reversal schedule $\tilde{S}^1$ needs no more than $k-2$ processors and checkpoints for the reversal of $l_{k-2}+1$ physical steps. This is a contradiction to the definition of l_{k-2}. Hence the inequality $l_{S^2} > \bar{t}\, l_{k-2} - \hat{t} + 1$ cannot be true. Therefore one has

$$l_k = l_{S^2} + l_{S^1} \leq \bar{t}\, l_{k-2} - \hat{t} + 1 + l_{k-1}$$

and the assertion is proven.

Second assume that $l_{S^1} \leq l_{k-2}$. If $l_{S^2} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - l_{S^1}$ one obtains

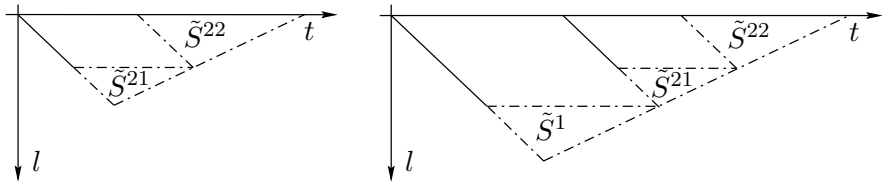
$$l_k = l_{S^2} + l_{S^1} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1$$

and the assertion is proven. If $l_{S^2} > l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - l_{S^1}$ it follows that

$$l_{S^2} \geq l_{k-1} + (\bar{t} - 1) l_{k-2} - \hat{t} + 2 \geq \bar{t}\, l_{S^1} - \hat{t} + 2\,. \tag{4.2}$$

This yields the inequality $l_{S^2} + \hat{t} - 1 \geq \bar{t}\, l_{S^1} + 1$. As before one obtains that S^1 needs no more than $k-2$ processors and checkpoints in the overlap. Furthermore, one finds that each computational cycle $\jmath = 1, \ldots, \bar{t}\, l_{S^1} + 1$ of S^1 needs no more than $k-2$ processors and checkpoints. It follows from Theorem 4.2 that there exists a feasible parallel schedule $\tilde{S}^1$ for the reversal of l_{S^1} physical steps such that $R(\tilde{S}^1) \leq R(S^1)$. Moreover no more than $k-2$ resources are needed in the last $l_{S^1} + \hat{t} - 1$ computational cycles of $\tilde{S}^1$. Therefore in the overlap and in the *front reverse* as depicted in Fig. 4.13 at most $k-2$ processors and checkpoints are used by $\tilde{S}^1$.

For $k \geq 3 + \lceil \hat{t}/\bar{t} \rceil$ follows that $l_{S^2} \geq 3$ because of Inequality (4.2). Hence it is possible to apply Theorem 4.1 to S^2. On this account there exists a parallel schedule $\tilde{S}^2$ to reverse l_{S^2} physical steps with the structure shown in Fig. 4.15 and $R(\tilde{S}^2) \leq R(S^2)$. Consider the parallel schedule $\tilde{S}$ displayed in Fig. 4.16 to reverse l_k physical steps. Here a checkpoint instead of a processor



Schedule $\tilde{S}^2$ Schedule $\tilde{S}$

Figure 4.15: Structure of $\tilde{S}^2$ and the Preliminary $\tilde{S}$

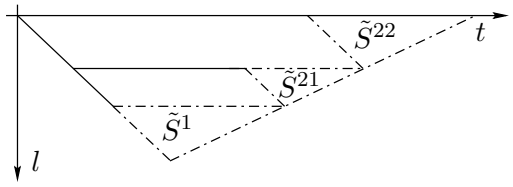


Figure 4.16: Structure of $\tilde{S}$

is used to start $\tilde{S}^{21}$. As before one has for each computational cycle of the *tail reverse* as depicted in Fig 4.13 and of the overlap that a maximal number of k resources are utilized because nothing has changed in these computational cycle apart from using a processor as a checkpoint. For the computational cycles of the front reverse no more than $k-2$ processors and checkpoints are needed in the corresponding computational cycles of $\tilde{S}^1$ as described above. In addition to these processors and checkpoints used by $\tilde{S}^1$ only two checkpoints are needed during the computational cycles of the front reverse. Hence one finds that the number of resources utilized in each computational cycle of the front reversal does not exceed k. In the remaining computational cycles of S that form the *start up* as depicted in Fig 4.13 the number of processors and checkpoints needed never exceeds 3, which is less than k. Hence the feasible parallel reversal schedule $\tilde{S}$ can be considered instead of S.

Let $l_{\tilde{S}^{21}}$ denote the number of physical steps reversed by $\tilde{S}^{21}$ and $l_{\tilde{S}^{22}}$ the number of physical steps reversed by $\tilde{S}^{22}$. One has to distinguish between the following two cases. First if $l_{S^1} + l_{\tilde{S}^{21}} > l_{k-2}$ or $l_{S^1} + l_{\tilde{S}^{21}} \leq l_{k-2}$ and $l_{\tilde{S}^{22}} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - (l_{S^1} + l_{\tilde{S}^{21}})$ the same argument as above proves the assertion. Second if $l_{S^1} + l_{\tilde{S}^{21}} \leq l_{k-2}$ and $l_{\tilde{S}^{22}} > l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - (l_{S^1} + l_{\tilde{S}^{21}})$ define $S \equiv \tilde{S}$, $S^2 \equiv \tilde{S}^{22}$, and $S^1 = \tilde{S}^1 + \tilde{S}^{21}$ as depicted in Fig. 4.17. Repeat the construction of $\tilde{S}$ as described above. This will yield that $l_{S^1} + l_{\tilde{S}^{21}} > l_{k-2}$

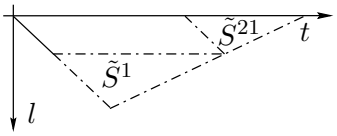


Figure 4.17: New Structure of S^1

or $l_{S^1} + l_{\tilde{S}^{21}} \leq l_{k-2}$ and $l_{\tilde{S}^{22}} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - l_{S^1}$ after at most $k-2$ consecutive constructions of $\tilde{S}$: The original S^1 employs at least one processor in the computational cycle $\jmath = \bar{t}\, l_{S^1} + 1$ of S^1, i.e. just after the vertex of S^1. In the first construction of the new S^1 as in Fig. 4.17 this number of resources needed increases by one because of the new checkpoint storing the new initial state. Hence after the first construction at least two processors and checkpoints are needed in the computational cycle $\jmath = \bar{t}\, l_{S_1^1} + 1$ of S_1^1. The new second index denotes the first construction. Applying this argumentation again and again one finds that at least three processors and checkpoints are used in the computational cycle $\jmath = \bar{t}\, l_{S_2^1} + 1$ of S_2^1, four processors and checkpoints are used in the computational cycle $\jmath = \bar{t}\, l_{S_3^1} + 1$ of S_3^1 and hence k processors and checkpoints are used in the computational cycle $\jmath = \bar{t}\, l_{S_{k-1}^1} + 1$ of S_{k-1}^1. This is a contradiction because together with the checkpoint storing the initial state 0 now $k+1$ processors and checkpoints are needed in the computational cycle $j = \bar{t}\, l_k + 1$ of S. This cannot be true. Therefore after no more than $k-2$ constructions one finds that $l_{S^1} + l_{\tilde{S}^{21}} > l_{k-2}$ or $l_{S^1} + l_{\tilde{S}^{21}} \leq l_{k-2}$ and $l_{\tilde{S}^{22}} \leq l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 - l_{S^1}$. Then the assertion can be proven using the same argument as above. ∎

An explicit formula has been proven for an upper bound of the maximal number l_k of physical steps that can be reversed with a feasible parallel reversal schedule using up to k processors and checkpoints at any time. In the following section it will be shown that this upper bound it also a lower bound and hence one has an equality in Equation (4.1).

4.3 Feasible Parallel Schedules to Reverse l_k Physical Steps

This section serves to construct feasible parallel reversal schedules that reverse l_k physical steps. In detail the following main theorem of this section will be proven in three subsections:

Theorem 4.4 (Feasible Parallel Reversal Schedules for $\hat{t}, \bar{t} \in \mathbb{N}$). *Suppose the temporal complexities $\hat{t} \in \mathbb{N}$ and $\bar{t} \in \mathbb{N}$ to perform one recording step and one reverse step, respectively, are given. The number of available resources each of which has the property of processor-checkpoint convertibility equals k. Then the upper bound*

$$l_k \;=\; \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} \tag{4.3}$$

of physical steps to be reversed with up to k resources at any time is actually attained by feasible parallel reversal schedules.

In order to construct feasible parallel schedules for reversing l_k physical steps one has to distinguish between three cases of temporal complexities of the reverse steps, namely $\bar{t} = 1$, $\bar{t} = 2$, and $\bar{t} > 2$. For each possible combination of $\bar{t}$ and $\hat{t}$ and all $k \in \mathbb{N}$ the same principle to develop the corresponding feasible parallel reversal schedules is used applying a recursive construction.

4.3.1 Feasible Parallel Reversal Schedules for $\bar{t} = 1$

First, the temporal complexities $\bar{t} = 1$ and $\hat{t} = 1$ will be examined. In this case one obtains the upper bound $l_k = l_{k-1}+l_{k-2}$ for $k \geq 3$ because of Equation (4.3). Therefore a natural approach in order to attain the upper bound would be to construct the feasible parallel reversal schedule S_k for $k \geq 3$ as composition $S_k = S_{k-2} + S_{k-1}$ of two feasible parallel schedules S_{k-1} and S_{k-2} for the reversal of l_{k-1} and l_{k-2} physical steps, respectively. The construction principle is illustrated by Fig. 4.18. Exactly this idea is used to prove Theorem 4.4 for

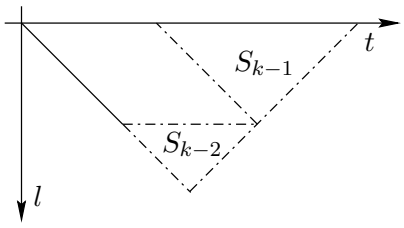


Figure 4.18: Reversal Schedule S_k

$\hat{t} = 1$ and $\bar{t} = 1$. For that purpose, the resource profile of S_k that results from the composition of S_{k-2} and S_{k-1} is analyzed in detail. Obviously, it would suffice to show that s_k^j is not greater than k.

Theorem 4.5 (Parallel Reversal Schedules for $\bar{t} = 1$ and $\hat{t} = 1$). *Suppose the given one-step evolution F determines the temporal complexities $\bar{t} = 1$ and $\hat{t} = 1$ to perform one reverse step and one recording step, respectively. Let k resources that can be used in each computational either as processor or as checkpoint be available. For the reversal of l_k physical steps, $k \geq 3$, there exist feasible parallel reversal schedules $S_k = S_{k-2}+S_{k-1}$ using up to k resources such that l_{k-2} and l_{k-1} physical steps are reversed by S_{k-2} and S_{k-1}, respectively. One obtains for the resource profile $R(S_k)$, $k > 3$,*

$$s_k^j = \begin{cases} i & for \quad l_{i-1} < j \leq l_i,\ 1 \leq i \leq k \\ k & for \quad j = l_k + 1 \\ k-i+1 & for \quad l_k + l_{i-1} < j \leq l_k + l_i,\ 2 \leq i \leq k-2 \\ 2 & for \quad l_k + l_{k-2} < j \leq 2l_k \end{cases} . \quad (4.4)$$

Proof. Consider the parallel reversal schedules to reverse l_k physical steps with $k = 1, 2, 3$ as shown in Fig. 4.19. For $k > 3$ the feasible parallel reversal schedule

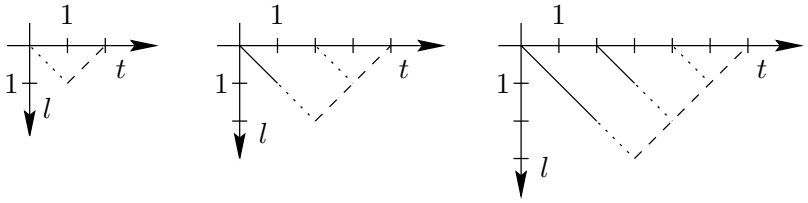


Figure 4.19: Feasible Parallel Reversal Schedules S_1, S_2, and S_3

S_k to reverse l_k physical steps is constructed recursively as composition of S_{k-2} and S_{k-1} as shown already in Fig. 4.18. Using an induction on k it will be proven that then S_k fulfils Equation (4.4). This behavior of s_k^j is depicted in Fig 4.20. The tickmarks are used to illustrated the different regions of Equation (4.4).

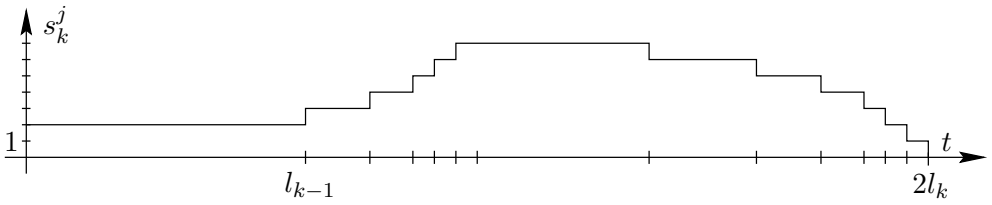


Figure 4.20: Behavior of $s_k^j \in R(S_k)$

For S_4 constructed according to Fig. 4.18 the corresponding resource profile $R(S_4)$ was already displayed in Fig. 4.4. It is easy to check that it fulfils exactly Equation (4.4). The feasible parallel reversal schedule $S_5 = S_3 + S_4$ and the resulting s_5^j are illustrated by Fig 4.21. The grey dotted line represents the

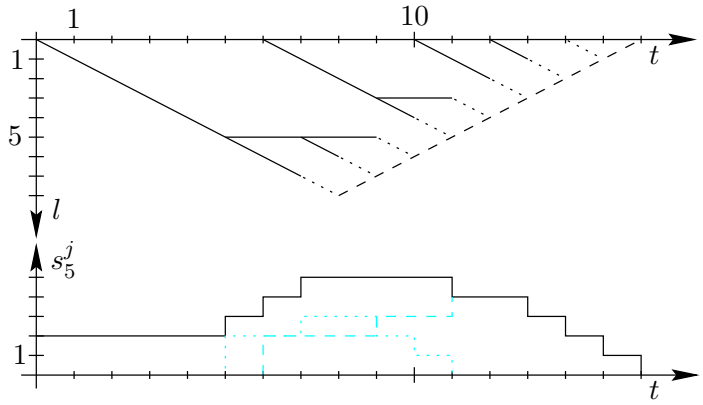


Figure 4.21: Reversal Schedule S_5 and $s_5^j \in R(S_5)$

profile s_3^j and the grey dashed line the profile s_4^j. Once more one can easily verify that Equation (4.4) hold exactly for $R(S_5)$.

Now it will be shown the following: If Equation (4.4) holds for the resource profiles of S_{k-1} and S_{k-2} then the resource profile of the parallel reversal schedule S_k constructed as the composition $S_{k-2} + S_{k-1}$ fulfils also Equation (4.4).

The parallel reversal schedule S_k takes the computational cycles j with $j = 1, \ldots, l_{k-1}$ over from of S_{k-1} without any changes. Therefore one has

$$s_k^j = s_{k-1}^j = i \quad \text{for} \quad l_{i-1} < j \leq l_i,\ i = 1, \ldots, k-1\ .$$

Now S_{k-2} is placed at the vertex of S_{k-1}. Hence one obtains for $j = l_{k-1} + 1$

$$s_k^{l_{k-1}+1} = s_{k-1}^{l_{k-1}+1} + s_{k-2}^1 = k - 1 + 1 = k\ .$$

Furthermore, it follows for $l_{i-1} < j \leq l_i$ with $i = 2, \ldots, k-3$ that

$$s_k^{l_{k-1}+j} = s_{k-1}^{l_{k-1}+j} + s_{k-2}^j \ = \ k - i + i = k$$

and for $l_{k-3} < j \leq l_{k-2}$, i.e. $i = k-2$, that

$$s_k^{l_{k-1}+j} = s_{k-1}^{l_{k-1}+j} + s_{k-2}^{j} \;=\; 2 + k - 2 = k \;.$$

This is possible because $l_{k-1} + l_{k-2} < 2\,l_{k-1}$. Now it is shown that the resource profile $R(S_k)$ fulfils Equation (4.4) for all computational cycles $j \leq l_k$. For the three following computational cycles follows

$$\begin{aligned} j &= l_k + 1: \quad s_k^{l_k+1} = s_{k-1}^{l_k+1} + s_{k-2}^{l_{k-2}+1} = 2 + k - 2 = k \\ j &= l_k + 2: \quad s_k^{l_k+2} = s_{k-1}^{l_k+2} + s_{k-2}^{l_{k-2}+2} = 2 + k - 3 = k - 1 \\ j &= l_k + 3: \quad s_k^{l_k+3} = s_{k-1}^{l_k+3} + s_{k-2}^{l_{k-2}+3} = 2 + k - 4 = k - 2 \end{aligned}$$

because $l_k + 3 \leq l_{k-1} + l_{k-2} + l_{k-3} = 2\,l_{k-1}$ for $k > 5$. With respect to S_{k-1} there are $2\,l_{k-1} - l_{k-1} - l_{k-2} - 3 = l_{k-3} - 3$ computational cycles left each of which needs only one checkpoint and only one processor. For $l_k + l_{i-1} < j \leq l_k + l_i$ and $i = 4, \ldots, k-4$ follows

$$s_k^j = s_{k-1}^j + s_{k-2}^{j-l_{k-1}} \;=\; 2 + k - i - 1 \;=\; k - i + 1$$

and for $l_k + l_{k-4} < j \leq l_k + l_{k-3}$, i.e. $i = k-3$,

$$s_k^j = s_{k-1}^j + s_{k-2}^{j-l_{k-1}} \;=\; 2 + 2 = k - (k-3) + 1 \;=\; k - i + 1 \;.$$

After taking all computational cycles of the parallel reversal schedule S_{k-1} into account from now on one checkpoint has to store the initial state 0. Therefore one finds for $l_k + l_{k-3} < j \leq l_k + l_{k-2}$, i.e. $i = k-2$, that

$$s_k^j = 1 + s_{k-2}^{j-l_{k-1}} \;=\; 1 + 2 = k - (k-2) + 1 \;=\; k - i + 1$$

because $j - l_{k-1} \leq l_k + l_{k-2} - l_{k-1} = 2\,l_{k-2}$. Hence also the parallel reversal schedule S_{k-2} is completed. Now one processor has to run from the initial state 0 to state l_{k-1} where the parallel reversal schedule S_{k-2} starts. Additionally one checkpoint stores the initial state 0. This yields

$$s_k^j = 2 \quad \text{for} \quad l_k + l_{k-2} < j \leq 2\,l_k \;.$$

It follows that $R(S_k)$ fulfils Equation (4.4), which completes the proof. ∎

Because of Theorem 4.5 there exist feasible parallel reversal schedules S_k for reversing l_k physical steps using up to k resources at any time if $(\hat{t}, \bar{t}\,) = (1, 1)$. Hence for the natural number pair $(\hat{t}, \bar{t}\,) = (1, 1)$ Theorem 4.4 is proven. Nevertheless, in the worst case it could happen that one needs really k processors to reverse l_k physical steps because each resource can be used as processor. In order to bound above also the number of processor needed the behaviour of c_k^j and p_k^j, respectively, is considered in the next corollary:

Corollary 4.3 (Resource Profile $R(S_k)$ for $\bar{t} = 1$ and $\hat{t} = 1$).
For the resource profiles $R(S_k)$ of the feasible parallel reversal schedules S_k constructed in Theorem 4.5 and $k > 3$ one obtains

$$\begin{array}{lll} & p_k^j \leq \lceil \frac{i+1}{2} \rceil & \text{for} \;\; l_{i-1} < j \leq l_i,\; 1 \leq i \leq k \\ c_k^j = \lceil \frac{k-1}{2} \rceil, & p_k^j = \lfloor \frac{k+1}{2} \rfloor & \text{for} \;\; j = l_k + 1 \\ c_k^j = \lceil \frac{k-i+1}{2} \rceil, & p_k^j = \lfloor \frac{k-i+1}{2} \rfloor & \text{for} \;\; l_k + l_{i-1} < j \leq l_k + l_i,\; 2 \leq i \leq k-2 \\ c_k^j = 1, & p_k^j = 1 & \text{for} \;\; l_k + l_{k-2} < j \leq 2\,l_k \;. \end{array} \tag{4.5}$$

Proof. Figure 4.22 illustrates the behavior of c_k^j and p_k^j belonging to $R(S_k)$ for $k > 3$. The black lines represent the equalities in (4.5), whereas the upper

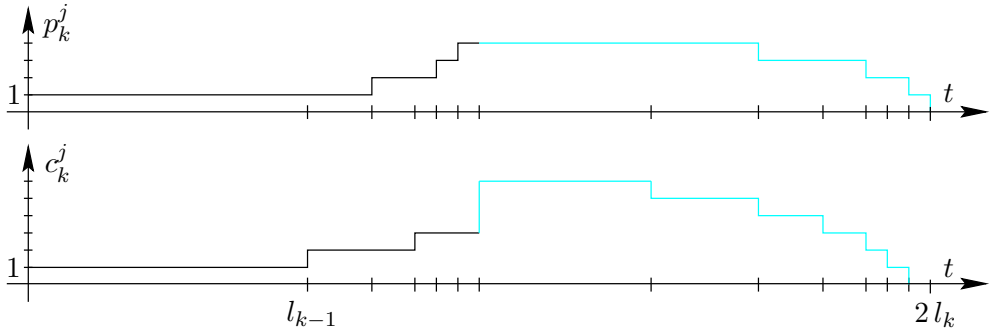


Figure 4.22: Behavior of $c_k^j, p_k^j \in R(S_k)$

bounds are illustrated as grey lines. The upper bound of c_k^j during the computational cycles $1 \leq j \leq l_k$ equals $s_k^j - 1$ because at least one processor performs one physical step, one recording steps or one reverse step.

For $S_4 = S_2 + S_3$ the behavior of c_k^j and p_k^j, respectively, was already displayed in Fig. 4.4. It is easy to check that Equation (4.5) holds. The feasible parallel reversal schedule S_5 constructed according to Fig. 4.18 and the resulting values of c_5^j and p_5^j are illustrated by Fig 4.23. One can conclude that they fulfil

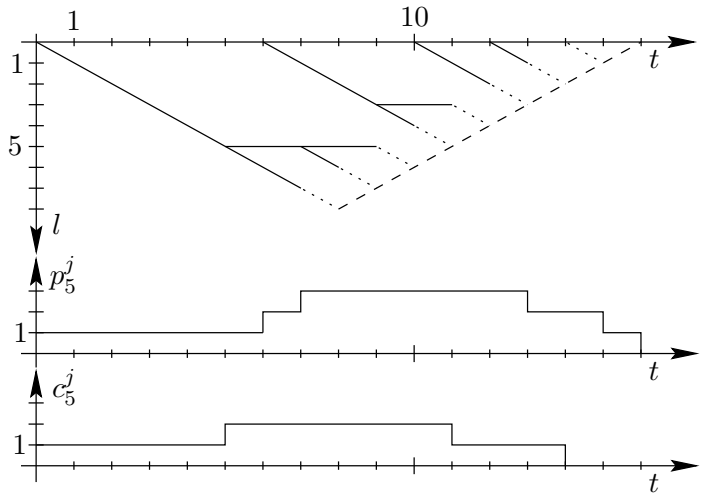


Figure 4.23: Reversal Schedule S_5 and $c_5^j, p_5^j \in R(S_5)$

also the requirements of Equation (4.5).

The remainder of this proof has the same structure as the proof of Theorem 4.5. Hence it will be shown the following: If Equation (4.5) holds for the resource profiles of S_{k-1} and S_{k-2} then the resource profile of the parallel reversal schedule S_k constructed as the composition $S_{k-2} + S_{k-1}$ fulfils also Equation (4.5).

The computational cycles $j = 1, \ldots, l_{k-1}$ of S_k and S_{k-1} are identical. Therefore it follows that

$$p_k^j = p_{k-1}^j \leq \left\lceil \frac{i+1}{2} \right\rceil \quad \text{for} \quad l_{i-1} < j \leq l_i,\ i = 1, \ldots, k-1\ .$$

The parallel reversal schedule S_{k-2} is placed at the vertex of S_{k-1}. Hence one has for $j = l_{k-1} + 1$

$$p_k^{l_{k-1}+1} = p_{k-1}^{l_{k-1}+1} + p_{k-2}^1 = \left\lfloor \frac{k}{2} \right\rfloor + 1 = \left\lceil \frac{k+1}{2} \right\rceil\ .$$

Moreover, one obtains for $l_{i-1} < j \leq l_i$ with $i = 2, \ldots, k-3$ that

$$p_k^{l_{k-1}+j} = p_{k-1}^{l_{k-1}+j} + p_{k-2}^j \ \leq\ \left\lfloor \frac{k-i}{2} \right\rfloor + \left\lceil \frac{i+1}{2} \right\rceil \leq \left\lceil \frac{k+1}{2} \right\rceil$$

and for $l_{k-3} < j \leq l_{k-2}$, i.e. $i = k-2$, that

$$p_k^{l_{k-1}+j} = p_{k-1}^{l_{k-1}+j} + p_{k-2}^j \ \leq\ 1 + \left\lceil \frac{k-1}{2} \right\rceil \leq \left\lceil \frac{k+1}{2} \right\rceil\ .$$

This is valid because $l_{k-1} + l_{k-2} < 2\, l_{k-1}$ holds. It is shown that S_k fulfils Equation (4.5) for all computational cycles $j \leq l_k$. For the next three computational cycles one finds

$$\begin{aligned}
j \ = \ l_k + 1: \quad & c_k^{l_k+1} = c_{k-1}^{l_k+1} + c_{k-2}^{l_{k-2}+1} = 1 + \left\lceil \frac{k-3}{2} \right\rceil = \left\lceil \frac{k-1}{2} \right\rceil, \\
& p_k^{l_k+1} = p_{k-1}^{l_k+1} + p_{k-2}^{l_{k-2}+1} = 1 + \left\lfloor \frac{k-1}{2} \right\rfloor = \left\lfloor \frac{k+1}{2} \right\rfloor \\
j \ = \ l_k + 2: \quad & c_k^{l_k+2} = c_{k-1}^{l_k+2} + c_{k-2}^{l_{k-2}+2} = 1 + \left\lceil \frac{k-3}{2} \right\rceil = \left\lceil \frac{k-1}{2} \right\rceil, \\
& p_k^{l_k+2} = p_{k-1}^{l_k+2} + p_{k-2}^{l_{k-2}+2} = 1 + \left\lfloor \frac{k-3}{2} \right\rfloor = \left\lfloor \frac{k-1}{2} \right\rfloor \\
j \ = \ l_k + 3: \quad & c_k^{l_k+3} = c_{k-1}^{l_k+3} + c_{k-2}^{l_{k-2}+3} = 1 + \left\lceil \frac{k-4}{2} \right\rceil = \left\lceil \frac{k-2}{2} \right\rceil, \\
& p_k^{l_k+3} = p_{k-1}^{l_k+3} + p_{k-2}^{l_{k-2}+3} = 1 + \left\lfloor \frac{k-4}{2} \right\rfloor = \left\lfloor \frac{k-2}{2} \right\rfloor
\end{aligned}$$

because $l_k + 3 \leq l_{k-1} + l_{k-2} + l_{k-3} = 2\, l_{k-1}$ for $k > 5$. With respect to S_{k-1} there are $2\, l_{k-1} - l_{k-1} - l_{k-2} - 3 = l_{k-3} - 3$ computational cycles left each of which needs only one checkpoint and only one processor. Hence it follows for $l_k + l_{i-1} < j \leq l_k + l_i$ and $i = 4, \ldots, k-4$ that

$$\begin{aligned}
c_k^j \ &= \ c_{k-1}^j + c_{k-2}^{j-l_{k-1}} \ = \ 1 + \left\lceil \frac{k-i-1}{2} \right\rceil \ = \ \left\lceil \frac{k-i+1}{2} \right\rceil, \\
p_k^j \ &= \ p_{k-1}^j + p_{k-2}^{j-l_{k-1}} \ = \ 1 + \left\lfloor \frac{k-i-1}{2} \right\rfloor \ = \ \left\lfloor \frac{k-i+1}{2} \right\rfloor
\end{aligned}$$

and for $l_k + l_{k-4} < j \leq l_k + l_{k-3}$, i.e. $i = k-3$, that

$$\begin{aligned}
c_k^j \ &= \ c_{k-1}^j + c_{k-2}^{j-l_{k-1}} \ = \ 1 + 1 \ = \ 2 \ = \ \left\lceil \frac{k-(k-3)+1}{2} \right\rceil \ = \ \left\lceil \frac{k-i+1}{2} \right\rceil, \\
p_k^j \ &= \ p_{k-1}^j + p_{k-2}^{j-l_{k-1}} \ = \ 1 + 1 \ = \ 2 \ = \ \left\lfloor \frac{k-(k-3)+1}{2} \right\rfloor \ = \ \left\lfloor \frac{k-i+1}{2} \right\rfloor .
\end{aligned}$$

The parallel reversal schedule S_{k-1} is completed. From now on one checkpoint has to store the initial state 0. Therefore one finds for $l_k + l_{k-3} < j \leq l_k + l_{k-2}$, i.e. $i = k-2$, that

$$\begin{aligned} c_k^j &= 1 + c_{k-2}^{j-l_{k-1}} = 2 = \left\lceil \frac{k-(k-2)+1}{2} \right\rceil = \left\lceil \frac{k-i+1}{2} \right\rceil, \\ p_k^j &= p_{k-2}^{j-l_{k-1}} = 1 = \left\lfloor \frac{k-(k-2)+1}{2} \right\rfloor = \left\lfloor \frac{k-i+1}{2} \right\rfloor \end{aligned}$$

because $j - l_{k-1} \leq l_k + l_{k-2} - l_{k-1} = 2\,l_{k-2}$ is valid. The parallel reversal schedule S_{k-2} is completed. One processor has to run from the initial state 0 to state l_{k-1} where the parallel reversal schedule S_{k-2} starts. Additionally one checkpoint stores the initial state 0. Therefore one has

$$c_k^j = 1 \quad \text{and} \quad p_k^j = 1$$

for $l_k + l_{k-2} < j \leq 2\,l_k$. It follows that Equation (4.5) is valid for S_k, which completes the proof. ∎

Assume that $k \geq 3$ resources are available and each resource can be used either as processor or as checkpoint. Suppose the given uniform one-step evolution determines the temporal complexities $\hat{t} = 1$ and $\bar{t} = 1$. Because of the Theorems 4.3 and 4.5 one can easily determine the maximal number of physical steps that can be reversed with k resources, namely $l_k = l_{k-1} + l_{k-2}$ with $l_1 = 1$ and $l_2 = 2$. Furthermore, starting with the feasible parallel reversal schedules shown in Fig. 4.19 an appropriate feasible parallel schedule for the reversal of l_k physical steps can be constructed recursively according to the rule $S_k = S_{k-2} + S_{k-1}$ for $k > 3$ as depicted already in Fig. 4.18. It should be noted that S_k is formed by optimal subschedules S_{k-1} and S_{k-2}. Because of Corollary 4.3 no more than $\lceil (k+1)/2 \rceil$ processors are needed during this reversal process. Furthermore, one can conclude from Theorem 4.5 that the number s_k^j of resources needed during the reversal increases monotonously until the vertex of S_k is reached. Subsequently it decreases monotonously. The maximal number k of resources is attained in the computational cycles before the vertex of S_k and in the computational cycle right after the vertex of S_k. In Corollary 4.3 it was shown that the upper bound of processors needed during the reversal grows monotonously before the vertex of S_k and declines monotonously after the vertex of S_k. A task left is to determine the distribution of the processors during the reversal process. This will be a subject of future work.

One will see that the feasible parallel reversal schedules S_k constructed for the case $\hat{t} = 1$ and $\bar{t} = 1$ form the base to construct appropriate feasible parallel reversal schedules for situations $\hat{t} \in \mathbb{N}$, $\hat{t} > 1$ and $\bar{t} = 1$. The combinations $(\hat{t}, \bar{t}) \in \mathbb{N} \times \mathbb{N}$ with $\hat{t} > \bar{t} = 1$ represent an important part of uniform one-step evolutions. As described already in the introduction of this chapter the parameter $\bar{t}$ has a big influence on the time needed to execute a feasible parallel reversal schedule. Therefore it is worthwhile to reduce $\bar{t}$ at the expense of $\hat{t}$ for example by using preaccumulation strategies. For all uniform one-step evolutions combining a reasonable number of physical steps it would be optimal to reduce $\bar{t}$ to 1 by a corresponding increase in $\hat{t}$, i.e. the temporal complexity of

the recording steps. In these cases the upper bound of the number of physical steps that can be reversed with up to k resources at any time is given by

$$l_k \equiv \begin{cases} k & \text{if} \quad k < 2 + \hat{t} \\ l_{k-1} + l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} .$$

Hence the composition of the feasible parallel reversal schedules S_{k-2} and S_{k-1} for reversing l_{k-2} and l_{k-1} physical steps does obviously not yield the desired feasible parallel schedule S_k for the reversal of l_k physical steps. Therefore another construction principle for the feasible parallel reversal schedules S_k has to be applied if the given one-step evolution determines the temporal complexities $\hat{t} > 1$ and $\bar{t} = 1$. One finds the following relation to the pair $(\hat{t}, \bar{t}) = (1, 1)$ considered before:

Lemma 4.4.
Let $\hat{t} \in \mathbb{N}$ with $\hat{t} > 1$ be given and define with $k \in \mathbb{N}$

$$l_k \equiv \begin{cases} k & \textit{if} \quad k < 2 + \hat{t} \\ l_{k-1} + l_{k-2} - \hat{t} + 1 & \textit{else} \end{cases} \qquad \textit{and}$$

$$l_{k,1} \equiv \begin{cases} k & \textit{if} \quad k \leq 2 \\ l_{k-1} + l_{k-2} & \textit{else} \end{cases} .$$

For $k \geq 2 + \hat{t}$ follows that

$$l_k = l_{k-\hat{t}+1,1} + \hat{t} - 1 . \tag{4.6}$$

Proof. In order to show Equation (4.6) an induction on k is used. First consider $k = 2 + \hat{t}$. According to the definitions of l_k and $l_{k,1}$ one obtains

$$\begin{aligned} &l_k = k - 1 + k - 2 - \hat{t} + 1 = 2k - \hat{t} - 2 = 2 + \hat{t}, \\ &l_{k-\hat{t}+1,1} = l_{3,1} = 3, \\ &l_{k+1} = 2 + \hat{t} + k - 1 - \hat{t} + 1 = k + 2 = 4 + \hat{t}, \quad \text{and} \\ &l_{k-\hat{t}+2,1} = l_{4,1} = 5 . \end{aligned}$$

Therefore one has $l_k = 2 + \hat{t} = l_{3,1} + \hat{t} - 1$ and $l_{k+1} = 4 + \hat{t} = l_{4,1} + \hat{t} - 1$. Assume that Equation (4.6) holds for $2 + \hat{t} \leq j \leq k - 1$ with $k > 3 + \hat{t}$. This yields

$$\begin{aligned} l_k &= l_{k-1} + l_{k-2} - \hat{t} + 1 \\ &= l_{k-\hat{t},1} + \hat{t} - 1 + l_{k-\hat{t}-1,1} + \hat{t} - 1 - \hat{t} + 1 \\ &= l_{k-\hat{t}+1,1} + \hat{t} - 1 . \end{aligned}$$

Hence, Equation (4.6) has been proven for all $k \in \mathbb{N}$. ∎

Furthermore, one can make the following observation. If $(\hat{t}, \bar{t}) = (1, 1)$ is valid at most one processor is used to perform a recording step. If one has $\hat{t} > 1$ and $\bar{t} = 1$ one needs up to $\hat{t}$ processors in order to perform the recording steps of one computational cycle. Hence, a first idea could be to modify the parallel reversal schedules constructed in the proof of Theorem 4.5 such that the value $\hat{t} > 1$ is taken into account. This will lead to feasible schedules for the reversal of $l_{k-\hat{t}+1}$ physical steps if k resources are available. Using the modified feasible parallel reversal schedules and Lemma 4.4 it is possible to derive feasible parallel schedules to reverse l_k physical steps as follows:

Theorem 4.6 (Parallel Reversal Schedules for $\bar{t} = 1$ and $\hat{t} > 1$). *Suppose the given uniform one-step evolution F determines the temporal complexities $\bar{t} = 1$ and $\hat{t} > 1$ to perform one reverse step and one recording step, respectively. Let k resources be available each of which can be used either as processor or as checkpoint. It is possible to construct feasible parallel schedules S_k for the reversal of l_k physical steps which need no more than k resources at any time on the base of the feasible parallel schedules developed in Theorem 4.5. The number of processors needed by S_k does not exceed*

$$p_k^b \equiv \begin{cases} k & \text{if } 1 \le k < 2 + \hat{t} \\ \lceil (k + \hat{t})/2 \rceil & \text{else} \end{cases} .$$

Proof. All feasible parallel schedules to reverse l_k physical steps need $2\,l_k + \hat{t} - 1$ computational cycles. First, assume that $1 \le k < 2 + \hat{t}$. For each value of k one possible parallel reversal schedule S_k consists of one processor performing the first forward sweep from the initial state 0 to the penultimate state $k - 1$, the recording step $\hat{F}_{k-1}$, and all reverse steps $\bar{F}_i$, $k > i \ge 0$. Furthermore, one processor serves as checkpoint storing the initial state. The same processor performs the recording step $\hat{F}_0$ during the computational cycles $\hat{t} + 1$ down to 2. Moreover each of the $k - 2$ processors left starts at the computational cycle $\hat{t} + 1 + 2i$, $i = 1, \ldots, k - 2$, in order to perform a forward sweep from the initial state 0 to state i and the recording step $\hat{F}_i$ in time. The resulting parallel reversal schedules are depicted in principle by Fig. 4.24.

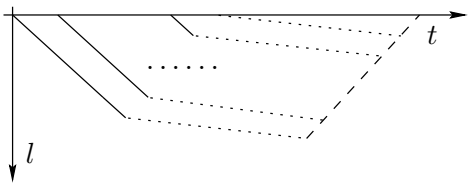


Figure 4.24: Parallel Reversal Schedule S_k, where $k < 2 + \hat{t}$

It is easy to check that this approach yields the resource profiles

$$\begin{array}{lll} c_k^j = 0,\ p_k^j = j, & s_k^j = j & \text{for } 1 \le j \le k \\ c_k^j = 0,\ p_k^j = k, & s_k^j = k & \text{for } k < j \le 1 + \hat{t} \\ c_k^j = 1,\ p_k^j = k - i,\ & s_k^j = k - i + 1 & \text{for } 2i - 2 < j - 1 - \hat{t} \le 2i,\ 1 \le i \le k - 1\,. \end{array}$$

Hence the assertion is proven for $1 \le k < 2 + \hat{t}$.

Second, for $k \ge 2 + \hat{t}$ feasible parallel reversal schedules S_k that need no more than k resources to reverse l_k physical steps are constructed. Furthermore, these S_k apply no more than p_k^b processors in each computational cycle. Because of Lemma 4.4 Equation (4.6) is valid. Using this equality appropriate parallel reversal schedules S_k will be constructed using the parallel reversal schedules $S_{k-\hat{t}+1,1}$ of Theorem 4.5 to reverse $l_{k-\hat{t}+1,1}$ physical steps. Set $\tilde{k} = k - \hat{t} + 1$ and consider for each $k \ge 2 + \hat{t}$ the first $l_{\tilde{k},1} + 1$ computational cycles of $S_{\tilde{k},1}$ as constructed in Theorem 4.5. Transform them into the first $l_{\tilde{k},1} + \hat{t}$ computational cycles of a parallel reversal schedule $\bar{S}_k$ such that the given value of

$\hat{t} > 1$ is taken into account. This corresponds to shifting the reverse sweep to the right as depicted in Fig. 4.25 for $k = 7$ and $\hat{t} = 3$. Everything else,

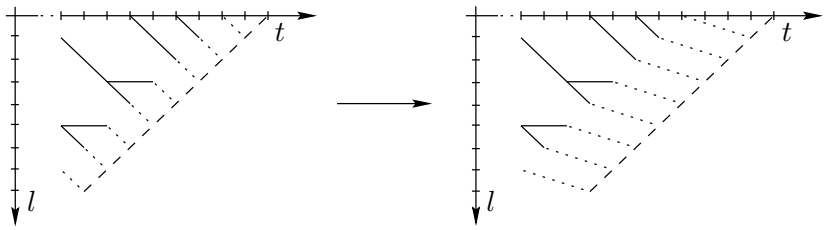


Figure 4.25: Modification of $S_{\tilde{k},1}$ for $k = 7$ and $\hat{t} = 3$

especially the checkpoint writings, stay the same. Therefore in each of these computational cycles $j > \hat{t}$ of $\bar{S}_k$ one obtains that $\hat{t}$ processors perform recording steps instead of one processor performing one recording step in $S_{\tilde{k},1}$. For that reason the resource profile of the first $l_{\tilde{k},1} + \hat{t}$ computational cycles of $\bar{S}_k$ can be derived as follows. The computational cycle $l_{\tilde{k},1} + 1$ of the parallel reversal schedule $S_{\tilde{k},1}$ constructed in Theorem 4.5 corresponds exactly to the computational cycle $l_{\tilde{k},1} + \hat{t}$ of $\bar{S}_k$. The $l_{\tilde{k},1}$th computational cycle of $S_{\tilde{k},1}$ differs from the $(l_{\tilde{k},1} + \hat{t} - 1)$th computational cycle of $\bar{S}_k$: In the former a reverse step and a recording step are perform by two processors. In the latter the reverse step is not performed but two recording steps. Hence the number of processors and therefore the total number of checkpoints and processors are the same. For $m = 1, \ldots, \hat{t} - 2$ the computational cycles $j = l_{\tilde{k},1} - \hat{t} + 1 + m$ of $S_{\tilde{k},1}$ need two processors for a recording step and a reverse step. Comparing this with the corresponding computational cycles $l_{\tilde{k},1} + m$ with $m = 1, \ldots, \hat{t} - 2$ of $\bar{S}_k$ it follows that $\hat{t} + 1 - m$ processors are needed only for the recording steps. No reverse step is performed yet. Therefore, the number of processors needed and the total number of checkpoints and processors increase by $\hat{t} - 1 - m$ in each computational cycle $l_{\tilde{k},1} + m$ with $m = 1, \ldots, \hat{t} - 2$ of $\bar{S}_k$. Furthermore, two processors perform the recording and reverse steps in the computational cycles $j = 4, \ldots, l_{\tilde{k},1} - \hat{t} + 1$ of $S_{\tilde{k},1}$. During the corresponding computational cycles $j = \hat{t} + 3, \ldots, l_{\tilde{k},1}$ of $\bar{S}_k$ instead of one $\hat{t}$ processors are needed to perform the appropriate number of recording steps. Hence the number of processors required and the total number of checkpoints and processors increase by $\hat{t} - 1$ in each of those computational cycles. The resource profile for the remaining computational cycles $j = 1, \ldots, \hat{t} + 2$ is obvious. In detail one obtains

$$
\begin{array}{lll}
p_k^j = j, & s_k^j = j & 1 \le j \le \hat{t} + 1 \\
p_k^j = \hat{t} + 1, & s_k^j = \hat{t} + 2 & j = \hat{t} + 2 \\
p_k^j = p_{\tilde{k},1}^{j-\hat{t}+1} + \hat{t} - 1, & s_k^j = s_{\tilde{k},1}^{j-\hat{t}+1} + \hat{t} - 1 & \hat{t} + 2 < j \le l_{\tilde{k},1} \\
p_k^j = p_{\tilde{k},1}^{j-\hat{t}+1} + \hat{t} - 1 - m, & s_k^j = s_{\tilde{k},1}^{j-\hat{t}+1} + \hat{t} - 1 - m & j = l_{\tilde{k},1} + m,\ 1 \le m \le \hat{t} - 1 \\
p_k^j = p_{\tilde{k},1}^{j-\hat{t}+1}, & s_k^j = s_{\tilde{k},1}^{j-\hat{t}+1} & j = l_{\tilde{k},1} + \hat{t}\,.
\end{array}
\tag{4.7}
$$

Therefore it is possible to derive for all $j = 1, \ldots, l_{\tilde{k},1}$ the following inequalities:

$$\begin{aligned} p_k^j &\le \max\left\{\hat{t}+1, \max_{4\le j\le l_{\tilde{k},1}}\left\{p_{\tilde{k},1}^{j-\hat{t}+1}+\hat{t}-1\right\}\right\} \\ &\le \max\left\{\hat{t}+1, \left\lceil\frac{k-\hat{t}+2}{2}\right\rceil+\hat{t}-1\right\} = \left\lceil\frac{k+\hat{t}}{2}\right\rceil, \\ s_k^j &\le \max\left\{\hat{t}+2, \max_{4\le j\le l_{\tilde{k},1}}\left\{s_{\tilde{k},1}^{i-\hat{t}+1}+\hat{t}-1\right\}\right\} \\ &\le \max\left\{\hat{t}+2, k-\hat{t}+1+\hat{t}-1\right\} = k\ . \end{aligned} \tag{4.8}$$

Furthermore, it follows for each $j = l_{\tilde{k},1} + m$ with $1 \le m \le \hat{t}-1$ that

$$\begin{aligned} p_k^j &= p_{\tilde{k},1}^{j-\hat{t}+1} + \hat{t} - 1 - m \le \left\lceil\frac{k+\hat{t}}{2}\right\rceil - m, \\ s_k^j &= s_{\tilde{k},1}^{j-\hat{t}+1} + \hat{t} - 1 - m \le k - m\ . \end{aligned} \tag{4.9}$$

Hence, in the $(l_{\tilde{k},1}+m)$th computational cycle there are m processors available. One can put the $l_{\tilde{k},1}+\hat{t}$ modified computational cycles of $\bar{S}_k$ and the computational cycles $j = l_{\tilde{k},1}+2, \ldots, 2\,l_{\tilde{k},1}$ of $S_{\tilde{k},1}$ together to form a complete parallel reversal schedule $\bar{S}_k$ reversing $l_{\tilde{k},1}$ physical steps for the given $\hat{t}$. This is possible because the computational cycle $l_{\tilde{k},1}+\hat{t}$ of $\bar{S}_k$ is equal to the computational cycle $l_{\tilde{k},1}+1$ of $S_{\tilde{k},1}$. The corresponding parallel reversal schedule $\bar{S}_7$ of the example shown in Fig. 4.25 is illustrated by Fig. 4.26. For the computational

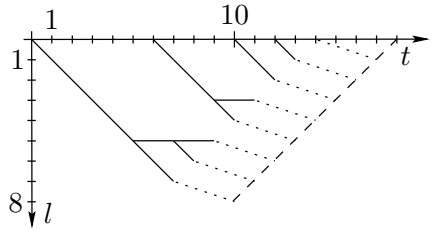


Figure 4.26: Complete Parallel Reversal Schedule $\bar{S}_7$

cycles $j = l_{\tilde{k},1}+\hat{t}+1, \ldots, 2\,l_{\tilde{k},1}+\hat{t}-1$ of $\bar{S}_k$ follows that

$$p_k^j \;=\; p_{\tilde{k},1}^{j-\hat{t}+1} \le \left\lceil\frac{k-\hat{t}+2}{2}\right\rceil \quad \text{and} \quad s_k^j \;=\; s_{\tilde{k},1}^{j-\hat{t}+1} \le k-\hat{t}+1\ . \tag{4.10}$$

Combining (4.9) and (4.10) one has for each $m = 1, \ldots, \hat{t}-1$ that one processor is available during the computational cycles $l_{\tilde{k},1}+m, \ldots, 2\,l_{\tilde{k},1}+\hat{t}-1$. This processor can be used the following way to increase the number of physical steps that are reversed and hence to create the desired parallel reversal schedule S_k. The free processor is applied to perform the reverse step $\bar{F}_{l_{\tilde{k},1}+m-1}$ in the computational cycle $l_{\tilde{k},1}+m$, the recording step $\hat{F}_{l_{\tilde{k},1}+m-1}$ during the

computational cycles $l_{\tilde{k},1}+m+1,\ldots,l_{\tilde{k},1}+m+\hat{t}$, and the forward sweep from the initial state 0 to the state $l_{\tilde{k},1}+m-1$ during the computational cycles $l_{\tilde{k},1}+m+\hat{t}+1,\ldots,2\,l_{\tilde{k},1}+2m+\hat{t}-1$. This extension is possible $\hat{t}-1$ times. The described construction principle is illustrated by Fig. 4.27 for the example considered above. Naturally, the checkpoint writing copying the initial state 0

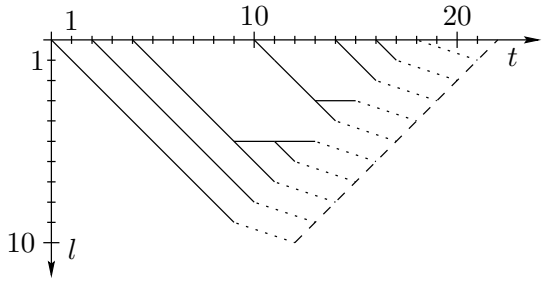


Figure 4.27: Resulting Parallel Reversal Schedule S_7

is performed already in the computational cycle $2\,l_k+\hat{t}-1$ instead of $2\,l_{\tilde{k},1}$. Hence because of Equation (4.6) the total number of physical steps that can be reversed with S_k is given by $l_{\tilde{k},1}+\hat{t}-1=l_k$. The number of computational cycles in S_k equals exactly $2\,l_{\tilde{k},1}+3\hat{t}-3=2\,l_k+\hat{t}-1$. Moreover, it follows from (4.8), (4.9), (4.10), and

$$\left\lceil\frac{k-\hat{t}+2}{2}\right\rceil+\hat{t}-1=\left\lceil\frac{k+\hat{t}}{2}\right\rceil$$

that the number of processors needed at any time is not greater than $\lceil\frac{k+\hat{t}}{2}\rceil$ and that the number of checkpoints and processors required does not exceed k, which completes the proof. ∎

Figure 4.27 shows the resulting parallel reversal schedule S_7 constructed building on the parallel reversal schedule $\bar{S}_7$ that can be seen in Fig. 4.26. Here, 7 resources are required. Furthermore, the number of processors applied does not exceed $5=\lceil\frac{k+\hat{t}}{2}\rceil$.

Suppose the given uniform one-step evolution F determines the temporal complexities $\hat{t}>1$ and $\bar{t}=1$, which is a desirable situation because of the overall run time needed for the reversal. Let k resources be available. For this case Theorem 4.6 proves the assertion of Theorem 4.4. Furthermore, it shows that one can reverse l_k physical steps on the base of the parallel reversal schedule $S_{k-\hat{t}+1}$ constructed in Theorem 4.5. One has to modify the recording steps to take $\hat{t}>1$ into account. This corresponds to shifting the reverse sweep to the right as illustrated by Fig. 4.25. Furthermore, one has to perform additional reversals of $\hat{t}-1$ physical steps at the vertex of $S_{k-\hat{t}+1}$ (see again Fig. 4.27). These modifications lead to a feasible parallel schedule for reversing l_k physical steps because of the equality $l_k=l_{k-\hat{t}+1,1}+\hat{t}-1$ proven in Lemma 4.4.

After the construction of feasible parallel reversal schedules for reversing l_k physical steps if one has $\hat{t}>1$ and $\bar{t}=1$ one can think about the minimization of the time the processors are used. One natural way to modify the

parallel reversal schedules developed in Theorem 4.6 is to perform the second checkpoint writing of $S_{\tilde{k},1}$ already in the forward sweep to state $l_k - 1$. Then the following $\hat{t} - 1$ forward sweeps can start at this checkpoint instead of the checkpoint storing the initial state. This strategy should be easy to use in an implementation of the parallel reversal schedules. Figure 4.28 shows the parallel reversal schedule using the modified checkpoint writing described in this paragraph for the example $k = 7$ and $\hat{t} = 3$. Nevertheless, it is not certain that this

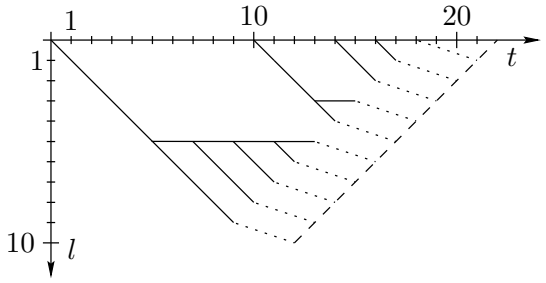


Figure 4.28: Modified Parallel Reversal Schedule for $k = 7$ and $\hat{t} = 3$

improvement, i.e. the minimization of the time the processors are used, can be exploited by the available multi-processor machines. One has to determine the number of processors required a priori. Often these processors are available for the complete reversal process. It is impossible to declare that they are needed only during a part of the computation.

At the end of this subsection the following conclusion can be drawn. Theorems 4.5 and 4.6 yield for one-step evolutions F with the temporal complexities $\bar{t} = 1$ and $\hat{t} \in \mathbb{N}$ feasible parallel reversal schedules to reverse l_k physical steps with no more than k processors and checkpoints, where l_k is given by

$$l_k \quad = \quad \begin{cases} k & \text{if} \quad 0 \leq k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} .$$

Furthermore, Theorem 4.3 shows that no more than l_k physical steps can be reversed with k processors and checkpoints. Therefore if one has the temporal complexities $\bar{t} = 1$ and $\hat{t} \in \mathbb{N}$ to perform one reverse step and one recording step, respectively, the maximal number of physical steps that can be reversed with k processors and checkpoints equals l_k.

4.3.2 Feasible Parallel Reversal Schedules for $\bar{t} = 2$

Suppose, the given one-step evolution F determines the temporal complexities $\bar{t} = 2$ and $\hat{t} \in \mathbb{N}$ to perform one reverse step and one recording step, respectively. More or less the same strategy as in the last subsection is used to construct feasible parallel schedules S_k for the reversal of l_k physical steps.

For $\hat{t} = 1$ and $k > 2$ one obtains the upper bound

$$l_k = l_{k-1} + 2\, l_{k-2}$$

with $l_1 = 1$ and $l_2 = 2$. Hence an obvious idea in order to develop parallel schedules S_k for reversing l_k physical steps would be to consider the composition $S_k = S_{k-2}+(S_{k-2}+S_{k-1})$. Here S_{k-1} and S_{k-2} are feasible parallel schedules for reversing l_{k-1} and l_{k-2} physical steps, respectively. This construction principle is depicted by Fig. 4.29. The following theorem constructs S_k according to the

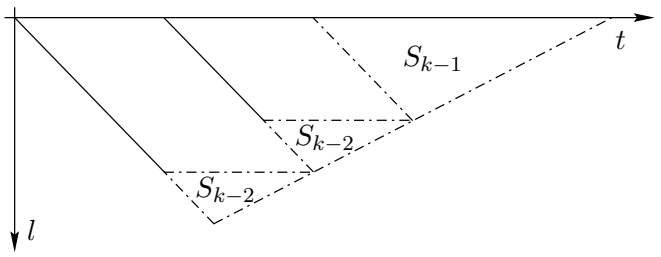


Figure 4.29: Parallel Reversal Schedule S_k for $k > 4$

rule $S_k = S_{k-2} + (S_{k-2} + S_{k-1})$ for $k > 4$ and analyzes the resulting resource profile $R(S_k)$. It will be shown that S_k needs no more than k resources each of which as the property of processor-checkpoint convertibility at any time.

Theorem 4.7 (Parallel Reversal Schedules for $\bar{t} = 2$ and $\hat{t} = 1$). *Suppose the given one-step evolution F determines the temporal complexities $\bar{t} = 2$ and $\hat{t} = 1$ to perform one reverse step and one recording step, respectively. Then there exist feasible parallel reversal schedules $S_k = S_{k-2} + (S_{k-2} + S_{k-1})$ for the reversal of l_k physical steps if $k > 4$, such that l_{k-1} and l_{k-2} physical steps can be reversed with the feasible parallel reversal schedules S_{k-1} and S_{k-2}, respectively. One obtains for the resource profile $R(S_k)$, $k \geq 4$,*

$$
\begin{array}{llll}
c_k^j = 0, & p_k^j = 1, & s_k^j = 1 & j = 1,2 \\
 & p_k^j \leq \lceil \frac{i+1}{2} \rceil, & s_k^j \leq i & l_i < j \leq l_{i+1},\ 2 \leq i \leq k \\
c_k^j = \lfloor \frac{k}{2} \rfloor, & p_k^j = \lceil \frac{k}{2} \rceil, & s_k^j = k & j = 2\,l_k + 1 \\
c_k^j = \lfloor \frac{k}{2} \rfloor, & p_k^j = \lfloor \frac{k-1}{2} \rfloor, & s_k^j = k-1 & 2\,l_k + 2 \leq j \leq 2\,l_k + 4 \\
c_k^j = \lfloor \frac{k-2}{2} \rfloor, & p_k^j = \lfloor \frac{k-2}{2} \rfloor, & s_k^j = 2\lfloor \frac{k-2}{2} \rfloor & \begin{cases} 2\,l_k+5 \leq j \leq 2\,l_k+l_m, \\ m = k - 2\lfloor \frac{k-4}{2} \rfloor \end{cases} \\
c_k^j = \lfloor \frac{k-i+2}{2} \rfloor, & p_k^j = \lfloor \frac{k-i+2}{2} \rfloor, & s_k^j = k-i+2 & \begin{cases} 2\,l_k+l_{i-1} < j \leq 2\,l_k+l_i, \\ i = m+2, m+4, \ldots, k\,. \end{cases}
\end{array}
\tag{4.11}
$$

Proof. For $k = 1$ and $k = 2$ the corresponding parallel reversal schedules are obvious. The parallel reversal schedule S_3 is depicted by Fig. 4.30. For $k = 4$ the

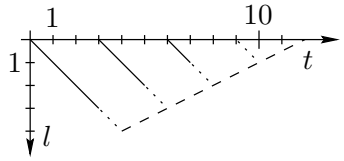


Figure 4.30: Parallel Reversal Schedule S_3 for $\bar{t} = 2$ and $\hat{t} = 1$

parallel reversal schedule S_4 as well as the resulting resource profile $R(S_4)$ are illustrated by Fig. 4.31. It can be checked easily that Equation (4.11) holds for

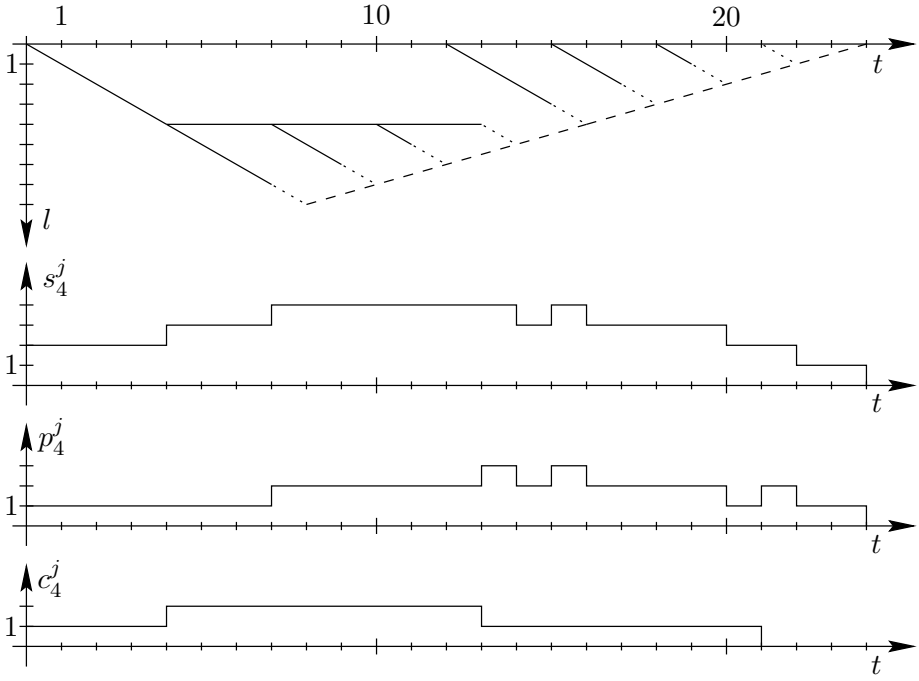


Figure 4.31: Parallel Reversal Schedule S_4 and Resource Profile $R(S_4)$

$R(S_4)$. In order to prove the assertion of the theorem also for all $k > 4$, parallel reversal schedules S_k to reverse l_k physical steps are constructed according to the procedure shown in Fig. 4.29.

For $S_5 = S_3 + (S_3 + S_4)$ one obtains the values of c_5^j, p_5^j, and s_5^j as illustrated by Fig. 4.32. Hence Equation (4.11) is valid for $R(S_5)$. Now assume that one has two parallel reversal schedules S_{k-2} and S_{k-1} the resource profiles of which satisfy Equation (4.11). One constructs $S_k = S_{k-2} + (S_{k-2} + S_{k-1})$ according to the procedure illustrated by Fig. 4.29. In the remainder of this proof it will be shown that the resulting parallel schedule S_k for the reversal of l_k physical steps fulfils Equation (4.11), too.

The parallel reversal schedule S_k takes the first $2\,l_{k-1}$ computational cycles over from S_{k-1} without any changes. Therefore one has for the number of checkpoints and the number of processors applied as well as for the sum of both during these computational cycles

$$c_k^j = c_{k-1}^j = 0, \;\; p_k^j = p_{k-1}^j = 1, \quad \text{and} \quad s_k^j = s_{k-1}^j = 1 \quad \text{for} \quad j = 1, 2$$

as well as

$$p_k^j = p_{k-1}^j \leq \left\lceil \frac{i+1}{2} \right\rceil \;\; \text{and} \;\; s_k^j = s_{k-1}^j \leq i \;\; \text{for} \;\; l_i < j \leq l_{i+1}, \; 2 \leq i \leq k-1 \; .$$

Then S_{k-2} is placed at the vertex of S_{k-1} so that the reversal can be performed

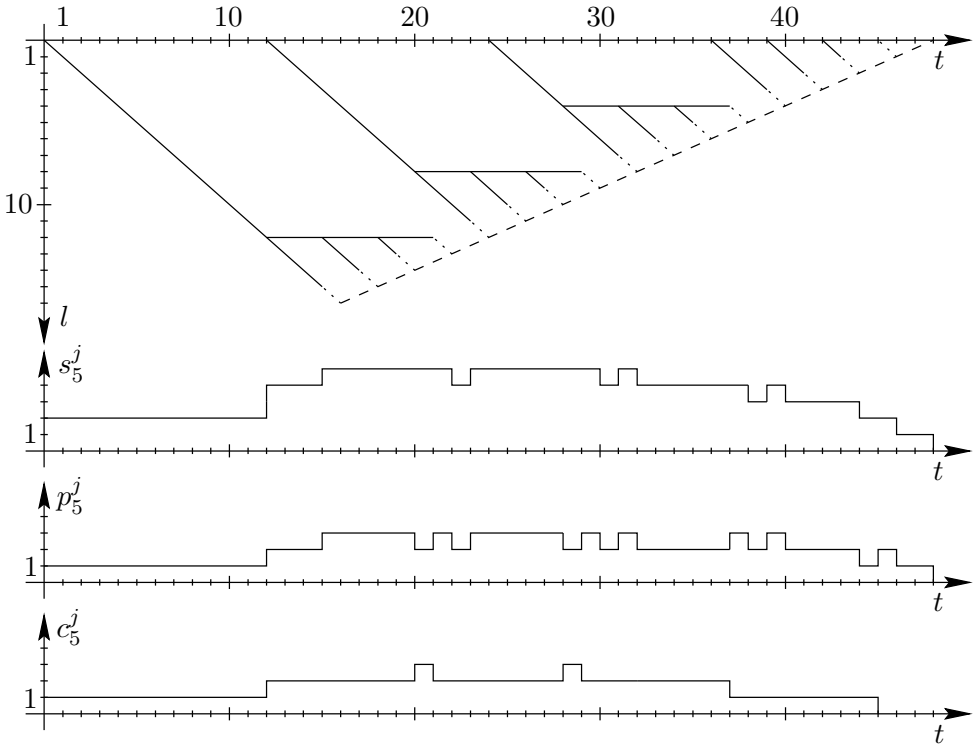


Figure 4.32: Parallel Reversal Schedule S_5 and Resource Profile $R(S_5)$

without any interruption. Hence for $j = l_k + 1$ follows that

$$p_k^j = p_{k-1}^j + p_{k-2}^1 = \left\lceil \frac{k-1}{2} \right\rceil + 1 = \left\lceil \frac{k+1}{2} \right\rceil, \quad s_k^j = s_{k-1}^j + s_{k-2}^1 = k \; . \tag{4.12}$$

Moreover one obtains for

$$\begin{aligned} j = l_k + 2: \quad & p_k^{l_k+2} = p_{k-1}^{l_k+2} + p_{k-2}^2 = \left\lfloor \frac{k-2}{2} \right\rfloor + 1 < \left\lceil \frac{k+1}{2} \right\rceil, \\ & s_k^{l_k+2} = s_{k-1}^{l_k+2} + s_{k-2}^2 = k - 2 + 1 < k \\ j = l_k + 3 \text{ and } j = l_k + 4: \quad & \\ & p_k^j = p_{k-1}^j + p_{k-2}^{j-l_k} \le \left\lfloor \frac{k-3}{2} \right\rfloor + 2 \le \left\lceil \frac{k+1}{2} \right\rceil, \\ & s_k^j = s_{k-1}^j + s_{k-2}^{j-l_k} \le k - 2 + 2 = k \end{aligned} \tag{4.13}$$

as well as for $j = l_k + 5, \ldots, l_k + l_{\tilde{m}}$ with $5 \ge \tilde{m} \equiv k - 1 - 2\lfloor \frac{k-5}{2} \rfloor \ge 4$

$$\begin{aligned} p_k^j &= p_{k-1}^j + p_{k-2}^{j-l_k} \le \left\lfloor \frac{k-3}{2} \right\rfloor + \left\lceil \frac{\tilde{m}}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= s_{k-1}^j + s_{k-2}^{j-l_k} \le 2\left\lfloor \frac{k-3}{2} \right\rfloor + \tilde{m} - 1 = k \; . \end{aligned} \tag{4.14}$$

It follows for $l_k + l_{i-1} < j \le l_k + l_i$, $i = \tilde{m} + 2, \tilde{m} + 4, \ldots, k - 3, k - 1$, that

$$\begin{aligned} p_k^j &= p_{k-1}^j + p_{k-2}^{j-l_k} \le \left\lfloor \frac{k-i+1}{2} \right\rfloor + \left\lceil \frac{i}{2} \right\rceil \le \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= s_{k-1}^j + s_{k-2}^{j-l_k} \le k - i + 1 + i - 1 = k \end{aligned} \tag{4.15}$$

because $l_k + l_i \leq l_k + l_{k-1} = 3\,l_{k-1}$. The parallel reversal schedule S_{k-1} is completed. From now on one has to take into account the checkpoint storing the initial state 0. The second parallel reversal schedule S_{k-2} is placed at the vertex of the first S_{k-2}. Therefore one obtains for

$$\begin{aligned}
j &= l_k + l_{k-1} + 1 = l_k + 2\,l_{k-2} + 1 : \\
&\quad p_k^{l_k+l_{k-1}+1} = p_{k-2}^{2\,l_{k-2}+1} + p_{k-2}^1 = \left\lceil \frac{k-2}{2} \right\rceil + 1 \leq \left\lceil \frac{k+1}{2} \right\rceil, \\
&\quad s_k^{l_k+l_{k-1}+1} = s_{k-2}^{2\,l_{k-2}+1} + s_{k-2}^1 + 1 = k - 2 + 1 + 1 = k \\
j &= l_k + l_{k-1} + 2 : \\
&\quad p_k^{l_k+l_{k-1}+2} = p_{k-2}^{2\,l_{k-2}+2} + p_{k-2}^2 = \left\lfloor \frac{k-3}{2} \right\rfloor + 1 < \left\lceil \frac{k+1}{2} \right\rceil, \\
&\quad s_k^{l_k+l_{k-1}+2} = s_{k-2}^{2\,l_{k-2}+2} + s_{k-2}^2 + 1 = k - 3 + 1 + 1 < k
\end{aligned} \tag{4.16}$$

as well as for $j = l_k + l_{k-1} + 3$ and $j = l_k + l_{k-1} + 4$

$$\begin{aligned}
p_k^j &= p_{k-2}^{j-l_k} + p_{k-2}^{j-l_k-l_{k-1}} \leq \left\lfloor \frac{k-3}{2} \right\rfloor + 2 \leq \left\lceil \frac{k+1}{2} \right\rceil, \\
s_k^j &= s_{k-2}^{j-l_k} + s_{k-2}^{j-l_k-l_{k-1}} + 1 \leq k - 3 + 2 + 1 = k\,.
\end{aligned} \tag{4.17}$$

For $j = l_k + l_{k-1} + 5, \ldots, l_k + l_{k-1} + l_{\hat{m}}$ with $5 \geq \hat{m} \equiv k - 2 - 2\lfloor \frac{k-6}{2} \rfloor \geq 4$ follows that

$$\begin{aligned}
p_k^j &= p_{k-2}^{j-l_k} + p_{k-2}^{j-l_k-l_{k-1}} \leq \left\lfloor \frac{k-4}{2} \right\rfloor + \left\lceil \frac{\hat{m}}{2} \right\rceil \leq \left\lceil \frac{k+1}{2} \right\rceil, \\
s_k^j &= s_{k-2}^{j-l_k} + s_{k-2}^{j-l_k-l_{k-1}} + 1 \leq 2\left\lfloor \frac{k-4}{2} \right\rfloor + \hat{m} - 1 + 1 = k\,.
\end{aligned} \tag{4.18}$$

For $l_k + l_{k-1} + l_{i-1} < j \leq l_k + l_{k-1} + l_i$ with $i = \hat{m} + 2, \hat{m} + 4, \ldots, k-4, k-2$ one has

$$\begin{aligned}
p_k^j &= p_{k-2}^{j-l_k} + p_{k-2}^{j-l_k-l_{k-1}} \leq \left\lfloor \frac{k-i}{2} \right\rfloor + \left\lceil \frac{i}{2} \right\rceil \leq \left\lceil \frac{k+1}{2} \right\rceil, \\
s_k^j &= s_{k-2}^{j-l_k} + s_{k-2}^{j-l_k-l_{k-1}} + 1 \leq k - i + i - 1 + 1 = k
\end{aligned} \tag{4.19}$$

because $j - l_k \leq l_{k-1} + l_{k-2} = 3\,l_{k-2}$. Now the first parallel reversal schedule S_{k-2} is completed. Therefore apart from the checkpoint storing the initial state 0 an additional processor is needed during the computational cycles j with $j = l_k + l_{k-1} + l_{k-2} + 1, \ldots, l_k + 2\,l_{k-1} + l_{k-2}$. It performs the forward sweep from the initial state 0 to state l_{k-1} in order to start the execution of the first parallel reversal schedule S_{k-2} in the computational cycle $l_k + l_{k-1} + l_{k-2}$. Hence for $j = l_k + l_{k-1} + l_{k-2} + 1, \ldots, l_k + l_{k-1} + 2\,l_{k-2} = 2\,l_k$ follows that

$$\begin{aligned}
p_k^j &= p_{k-2}^{j-l_k-l_{k-1}} + 1 \leq \left\lceil \frac{i+1}{2} \right\rceil + 1 = \left\lceil \frac{i+3}{2} \right\rceil \quad \text{and} \\
s_k^j &= s_{k-2}^{j-l_k-l_{k-1}} + 2 \leq i + 2
\end{aligned} \tag{4.20}$$

with $i = 2, \ldots, k-2$. Combining Equations (4.12) up to (4.20) it is shown that

$$p_k^j \leq \lceil \tfrac{k+1}{2} \rceil \quad \text{and} \quad s_k^j \leq k \quad \text{for} \quad l_k < j \leq l_{k+1}\,.$$

Furthermore, one has for $j = 2\,l_k + 1$

$$\begin{aligned} c_k^j &= c_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-2}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k}{2} \Big\rfloor, \\ p_k^j &= p_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lceil \frac{k-2}{2} \Big\rceil + 1 = \Big\lceil \frac{k}{2} \Big\rceil, \\ s_k^j &= s_{k-2}^{j-l_k-l_{k-1}} + 2 = k - 2 + 2 = k \end{aligned} \tag{4.21}$$

and for $j = 2\,l_k + 2,\ 2\,l_k + 3,\ 2\,l_k + 4$

$$\begin{aligned} c_k^j &= c_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-2}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k}{2} \Big\rfloor, \\ p_k^j &= p_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-3}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k-1}{2} \Big\rfloor, \\ s_k^j &= s_{k-2}^{j-l_k-l_{k-1}} + 2 = k - 3 + 2 = k - 1\ . \end{aligned} \tag{4.22}$$

Since $\hat{m} = k - 2 - 2\lfloor \frac{k-6}{2} \rfloor = k - 2\lfloor \frac{k-4}{2} \rfloor \equiv m$ one finds for the computational cycles $j = l_k + l_{k-1} + 5, \ldots, l_k + l_{k-1} + l_m$ that

$$\begin{aligned} c_k^j &= c_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-4}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k-2}{2} \Big\rfloor, \\ p_k^j &= p_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-4}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k-2}{2} \Big\rfloor, \\ s_k^j &= s_{k-2}^{j-l_k-l_{k-1}} + 2 = 2\Big\lfloor \frac{k-4}{2} \Big\rfloor + 2 = 2\Big\lfloor \frac{k-2}{2} \Big\rfloor\ . \end{aligned} \tag{4.23}$$

Moreover it follows for the computational cycles $2\,l_k + l_{i-1} < j \leq 2\,l_k + l_i$ with $i = m + 2, m + 4, \ldots, k - 4, k - 2$

$$\begin{aligned} c_k^j &= c_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-i}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k-i+2}{2} \Big\rfloor, \\ p_k^j &= p_{k-2}^{j-l_k-l_{k-1}} + 1 = \Big\lfloor \frac{k-i}{2} \Big\rfloor + 1 = \Big\lfloor \frac{k-i+2}{2} \Big\rfloor, \\ s_k^j &= s_{k-2}^{j-l_k-l_{k-1}} + 2 = k - i + 2 \end{aligned} \tag{4.24}$$

because $j \leq 2\,l_k + l_{k-2} = l_k + 2\,l_{k-1} + l_{k-2}$. The second parallel reversal schedule S_{k-2} is completed. During the remaining computational cycles j with $j = 2\,l_k + l_{k-2} + 1, \ldots, 3\,l_k$ only one checkpoint stores the initial state 0 and one processor performs a forward sweep from the initial state 0 to state $l_{k-1} + l_{k-2}$. Hence, one has for $2\,l_k + l_{k-2} < j \leq 3\,l_k$

$$c_k^j = 1 = \Big\lfloor \frac{k-i+2}{2} \Big\rfloor, \quad p_k^j = 1 = \Big\lfloor \frac{k-i+2}{2} \Big\rfloor, \quad \text{and} \quad s_k^j = k - i + 2 \tag{4.25}$$

with $i = k$. Equations (4.21) up to (4.25) show that S_k fulfils also the remainder of Equation (4.11), which completes the proof. ∎

Theorem 4.7 proves that the assertion of Theorem 4.4 is true for the temporal complexities $\hat{t} = 1$ and $\bar{t} = 2$. Furthermore, it yields a detailed description how to construct the desired feasible parallel reversal schedules. One has to use the recursion $S_k = S_{k-2} + (S_{k-2} + S_{k-1})$ for $k \geq 5$ as depicted in Fig. 4.29

with S_3 and S_4 defined as in Fig. 4.30 and Fig. 4.31, respectively. It should be noted that again S_k is formed by optimal subschedules S_{k-1} and S_{k-2}. As for the pair $(\hat{t}, \bar{t}\,) = (1, 1)$ the upper bound of the number of resources used in each computational cycle grows monotonously until the vertex of S_k is reached. Subsequently it declines monotonously. The maximal number k of resources is attained at least in the computational cycle $j = 2\,l_k + 1$ right after the vertex of S_k. The upper bound of the processors used increases monotonously before the vertex of S_k is reached and decreases monotonously in the remainder of the reversal process.

Now the temporal complexities $(\hat{t}, \bar{t}\,) = (2, 2)$ will be considered. One obtains for the upper bound l_k according to Equation (4.3) the formula

$$l_k \;=\; \begin{cases} k & \text{if} \quad k < 3 \\ l_{k-1} + 2\,l_{k-2} - 1 & \text{else} \end{cases} \quad .$$

This yields $l_k = 2\,l_{k-1}$ if k is even and $l_k = 2\,l_{k-1} - 1$ if k is odd. Therefore a first approach would be to construct the desired parallel reversal schedule S_k for reversing l_k physical steps as the composition $S_k = S_{k-1} + S_{k-1}$ if k is even. Here, S_{k-1} is a feasible parallel schedule for the reversal of l_{k-1} physical steps. If k is odd, one could try to form S_k as the composition $S_k = S_{k-1} + \bar{S}_{k-1}$. Here S_{k-1} is again a feasible parallel schedule for the reversal of l_{k-1} physical steps whereas $\bar{S}_{k-1}$ is a feasible parallel schedule for the reversal of $l_{k-1} - 1$ physical steps. A slightly different idea how to construct S_k is depicted in Fig. 4.33 using the feasible parallel reversal schedules S_{k-2} and $\bar{S}_{k-2}$ for the reversal of l_{k-2} and $l_{k-2} - 1$ physical steps. In order to create $\bar{S}_{k-2}$ the parallel reversal schedule

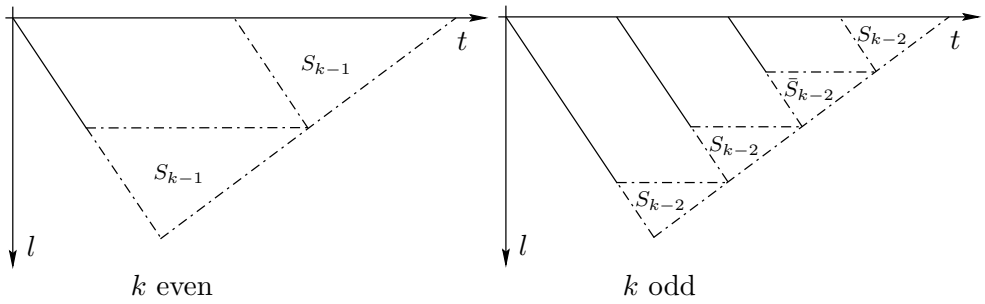


Figure 4.33: Parallel Reversal Schedule S_k

S_{k-2} is shortened by the reversal of the last physical step $F_{l_{k-2}-1}$. Exactly the construction principle illustrated in Fig. 4.33 is exploited by the next theorem. The resulting resource profiles are analyzes in order to prove that S_k needs no more that k resources each of which has the property of processor-checkpoint convertibility for the reversal of l_k physical steps.

Theorem 4.8 (Parallel Reversal Schedules for $\bar{t} = 2$ and $\hat{t} = 2$).
Suppose the given one-step evolution F determines the temporal complexities $\bar{t} = 2$ and $\hat{t} = 2$ to perform one reverse step and one recording step, respectively. A feasible parallel schedule S_k, $k \geq 4$, for the reversal of l_k physical steps can

be constructed according to

$$S_k \equiv \begin{cases} S_{k-1} + S_{k-1} & \text{if } k \text{ is even} \\ S_{k-2} + (S_{k-2} + (\bar{S}_{k-2} + S_{k-2})) & \text{if } k \text{ is odd} \end{cases}, \tag{4.26}$$

where S_{k-1}, S_{k-2}, and $\bar{S}_{k-2}$ are feasible parallel reversal schedules for reversing l_{k-1}, l_{k-2}, and $l_{k-2}-1$ physical steps, respectively. For the resource profile $R(S_k)$, $k>4$, follows

$$\begin{array}{llll}
c_k^j = 0, & p_k^j = 1, & s_k^j = 1 & j = 1,2 \\
 & p_k^j \le \lceil \frac{i+1}{2} \rceil, & s_k^j \le i & \begin{cases} l_i < j \le l_{i+1}+1, \ i \ \text{even} \\ l_i+1 < j \le l_{i+1}, \ i \ \text{odd} \\ i = 2,\ldots,k \end{cases} \\
c_k^j = \lfloor \frac{k}{2} \rfloor, & p_k^j = \lceil \frac{k}{2} \rceil, & s_k^j = k & j = 2\,l_k+1 \\
c_k^j = \lfloor \frac{k}{2} \rfloor, & p_k^j = \lfloor \frac{k-1}{2} \rfloor, & s_k^j = k-1 & j = 2\,l_k+2, 2\,l_k+3 \\
c_k^j = \lfloor \frac{k}{2} \rfloor, & p_k^j = \lfloor \frac{k-3}{2} \rfloor, & s_k^j = k-2 & j = 2\,l_k+4 \\
c_k^j = \lfloor \frac{k-i+3}{2} \rfloor, & p_k^j \le \lfloor \frac{k-i+2}{2} \rfloor, & s_k^j \le k-i+2 & \begin{cases} l_{i-2} < j-2\,l_k-1 \le l_i \\ i = 5,7,\ldots,2\lfloor \frac{k}{2} \rfloor - 1 \end{cases} \\
c_k^j = 1, & p_k^j = 1, & s_k^j = 2 & 2\,l_k + l_{2\lfloor \frac{k}{2} \rfloor - 1} < j-1 \le 3\,l_k.
\end{array} \tag{4.27}$$

Proof. See Appendix B, Page 115. ∎

In the proof of Theorem 4.8 the analyse of the resource profile resulting from the composition of feasible parallel reversal schedules forms again the main ingredient. The starting point is given by the schedules S_3 and S_4 for the reversal of $l_3 = 3$ and $l_4 = 6$ physical steps as shown in Fig. 4.34. Subsequently feasible

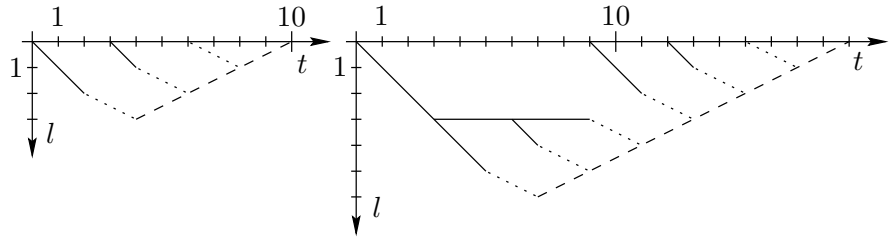


Figure 4.34: Parallel Reversal Schedules S_3 and S_4 for $\bar{t} = 2$ and $\hat{t} = 2$

parallel reversal schedules S_k are constructed according to Equation (4.26). It is shown that Equation (4.27) holds for $R(S_k)$ if $k > 4$. For that purpose also the resource profile of $\bar{S}_{k-2}$ is derived if k is odd. The proof of Theorem 4.8 is really technical and does not provide further insights. Therefore it is not presented here, but contained in Appendix B.

Theorem 4.8 implies that the assertion of Theorem 4.4 is valid for $(\hat{t}, \bar{t}\,) = (2,2)$. From the resource profile $R(S_k)$ one can conclude that the upper bound of resources used as well as of the processors needed increase monotonously before the vertex of S_k is reached and decrease monotonously afterwards. Once more the maximal number k of resources is needed at least in the computational cycle $j = 2\,l_k+1$ right after the vertex of S_k. Moreover, one should note the following

fact: The parallel reversal schedule S_k for odd k is built on a feasible reversal schedule $\bar{S}_{k-2}$ that is not optimal, i.e. it cannot be used for the reversal of l_{k-2} physical steps. This is not true for the combinations of $\hat{t}$ and $\bar{t}$ considered before.

If the given uniform one-step evolution F determines the temporal complexities $\hat{t} > 2$ and $\bar{t} = 2$ the upper bound l_k equals

$$l_k \equiv \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/2 \\ l_{k-1} + 2\,l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} .$$

One finds the following relation to the pairs $(\hat{t}, \bar{t}\,) = (1, 2)$ and $(\hat{t}, \bar{t}\,) = (2, 2)$ considered so far in this subsection:

Lemma 4.5.
Let $\hat{t} \in \mathbb{N}$ with $\hat{t} > 2$ be given and define with $k \in \mathbb{N}$

$$\begin{aligned} l_k &\equiv \begin{cases} k & \textit{if} \quad k < 2 + \hat{t}/2 \\ l_{k-1} + 2\,l_{k-2} - \hat{t} + 1 & \textit{else} , \end{cases} \\ l_{k,1} &\equiv \begin{cases} k & \textit{if} \quad k \le 2 \\ l_{k-1} + 2\,l_{k-2} & \textit{else} , \end{cases} \qquad \textit{and} \\ l_{k,2} &\equiv \begin{cases} k & \textit{if} \quad k \le 2 \\ l_{k-1} + 2\,l_{k-2} - 1 & \textit{else} \end{cases} . \end{aligned}$$

For $k \ge 2 + \hat{t}/2$ follows that

$$l_k = \begin{cases} l_{k-(\hat{t}-1)/2,1} + (\hat{t}-1)/2 & \textit{if } \hat{t} \textit{ is odd} \\ l_{k-(\hat{t}-2)/2,2} + (\hat{t}-2)/2 & \textit{if } \hat{t} \textit{ is even} \end{cases} . \tag{4.28}$$

Proof. In order to show Equation (4.28) consider first odd $\hat{t}$ and $k = 2+(\hat{t}+1)/2$. According to the definition of l_k and $l_{k,1}$, respectively, one has

$$\begin{aligned} &l_k = k - 1 + 2(k-2) - \hat{t} + 1 = 3k - \hat{t} - 4 = 3 + (\hat{t}+1)/2, \\ &l_{k-(\hat{t}-1)/2,1} = l_{3,1} = 4, \\ &l_{k+1} = 3 + (\hat{t}+1)/2 + 2(k-1) - \hat{t} + 1 = 7 + (\hat{t}+1)/2, \quad \text{and} \\ &l_{k+1-(\hat{t}-1)/2,1} = l_{4,1} = 8 . \end{aligned}$$

Therefore the equalities $l_k = 4 + (\hat{t}-1)/2 = l_{k-(\hat{t}-1)/2,1} + (\hat{t}-1)/2$ as well as $l_{k+1} = 8 + (\hat{t}-1)/2 = l_{k+1-(\hat{t}-1)/2,1} + (\hat{t}-1)/2$ are valid. Assume that Equation (4.28) holds for $2 + (\hat{t}+1)/2 \le j \le k-1$ with $k > 3 + (\hat{t}+1)/2$. Then one obtains

$$\begin{aligned} l_k &= l_{k-1} + 2\,l_{k-2} - \hat{t} + 1 \\ &= l_{k-1-(\hat{t}-1)/2,1} + (\hat{t}-1)/2 + 2\left(l_{k-2-(\hat{t}-1)/2,1} + (\hat{t}-1)/2\right) - \hat{t} + 1 \\ &= l_{k-(\hat{t}-1)/2,1} + (\hat{t}-1)/2 . \end{aligned}$$

For even $\hat{t}$ and $k = 2 + \hat{t}/2$ one finds according to the definitions of l_k and $l_{k,2}$

$$\begin{aligned} &l_k = k - 1 + 2(k-2) - \hat{t} + 1 = 3k - \hat{t} - 4 = 2 + \hat{t}/2, \\ &l_{k-(\hat{t}-2)/2,2} = l_{3,2} = 3, \\ &l_{k+1} = 2 + \hat{t}/2 + 2(k-1) - \hat{t} + 1 = 5 + \hat{t}/2, \quad \text{and} \\ &l_{k+1-(\hat{t}-2)/2,2} = l_{4,2} = 6 . \end{aligned}$$

This yields $l_k = l_{3,2} + (\hat{t}-2)/2$ and $l_{k+1} = l_{4,2} + (\hat{t}-2)/2$. Providing that Equation (4.28) holds for $2 + \hat{t}/2 \leq j \leq k-1$ with $k > 3 + \hat{t}/2$ one has

$$\begin{aligned} l_k &= l_{k-1} + 2\, l_{k-2} - \hat{t} + 1 \\ &= l_{k-1-(\hat{t}-2)/2,2} + (\hat{t}-2)/2 + 2\left(l_{k-2-(\hat{t}-2)/2,2} + (\hat{t}-2)/2\right) - \hat{t} + 1 \\ &= l_{k-(\hat{t}-1)/2,2} + (\hat{t}-2)/2\ . \end{aligned}$$

Therefore, Equation (4.28) is valid for all $\hat{t} > 2$ and $k \geq 2 + \hat{t}/2$. ∎

Having Equation (4.28) at hand the reversal schedules constructed in the Theorems 4.7 and 4.8 are applied to develop feasible reversal schedules S_k if the given uniform one-step evolution F determines temporal complexities $\bar{t} = 2$ and $\hat{t} > 2$. For that purpose the value $\hat{t} > 2$ is taken into account, which corresponds to shifting the reverse sweep to the right. Subsequently additional reversals of physical steps are performed at the vertex of the modified parallel reversal schedule. This leads to the following assertion:

Theorem 4.9 (Parallel Reversal Schedules for $\bar{t} = 2$ and $\hat{t} > 2$).
Suppose the given uniform one-step evolution F determines the temporal complexities $\bar{t} = 2$ and $\hat{t} > 2$ to perform one reverse step and one recording step, respectively. Let k resources be available each of which has the property of processor-checkpoint convertibility. If k is even one can construct a feasible parallel schedule S_k needing up to k resources at any time for the reversal of l_k physical steps on the base of the parallel schedules developed in Theorem 4.8. Otherwise a feasible parallel schedule S_k using up to k resources at any time for the reversal of l_k physical steps can be built on the parallel schedules developed in Theorem 4.7. Furthermore the number of processors applied by S_k does not exceed

$$p_k^b \equiv \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/2 \\ \lceil (k+1)/2 \rceil + \lceil (\hat{t}-1)/4 \rceil & \text{else} \end{cases}\ .$$

Proof. See Appendix B, Page 125. ∎

The proof consists of two main parts. In the first one S_k is constructed for odd $\hat{t}$. In the second one an appropriate parallel reversal schedule is developed for even $\hat{t}$. In both parts a similar argument as in the proof of Theorem 4.6 is utilized. Therefore the proof of Theorem 4.9 is not presented here, but contained in Appendix B.

Because of Theorem 4.9 the assertion of Theorem 4.4 is valid for $\hat{t} > 2$ and $\bar{t} = 2$. If $\hat{t}$ is odd the parallel reversal schedule $S_{k-(\hat{t}-1)/2,1}$ constructed in Theorem 4.7 is modified such that the value of $\hat{t}$ is taken into account. Hence the reverse sweep is shifted to the right. Furthermore it is shown in the proof of Theorem 4.9 that the reversal of $(\hat{t}-1)/2$ physical steps can be attached at the vertex of $S_{k-(\hat{t}-1)/2,1}$ according to the procedure depicted in Fig. 4.27. This yields the desired parallel reversal schedule because of Equation (4.28). If $\hat{t}$ is even $S_{k-(\hat{t}-2)/2,2}$ constructed in Theorem 4.8 forms the base of the desired parallel reversal schedule S_k. Once more one has to shift the reverse sweep

to the right in order to take the value of $\hat{t}$ into account. Moreover, in the proof of Theorem 4.9 it is shown that $(\hat{t}-2)/2$ physical steps can be reversed additionally at the vertex of $S_{k-(\hat{t}-2)/2,2}$ according to the procedure depicted in Fig. 4.27. Then the parallel reversal schedule S_k is constructed completely because of Lemma 4.5.

4.3.3 Feasible Parallel Reversal Schedules for $\bar{t} > 2$

Finally assume that the given uniform one-step evolution determines temporal complexities $\hat{t} \in \mathbb{N}$ and $\bar{t} > 2$ to perform one recording step and one reverse step, respectively. For $\hat{t} \leq \bar{t}$ appropriate parallel reversal schedules S_k will be constructed recursively. Then these parallel reversal schedules form the base to develop the desired S_k if $\hat{t} > \bar{t}$ is valid.

For the pair $(\hat{t}, \bar{t})$ with $\hat{t} \leq \bar{t}$ the upper bound l_k is defined as

$$l_k \;=\; \begin{cases} k & \text{if} \quad k \leq 2 \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} .$$

In order to construct appropriate S_k for reversing l_k physical steps one part of the composition could be represented by the parallel schedule S_{k-1} for reversing l_{k-1} physical steps. One finds that $\bar{t}\, l_{k-2} - \hat{t} + 1 > (\bar{t}-1) l_{k-2} > l_{k-2}$ is valid for $k \geq 3$. Hence, it is clear that the other part of the composition cannot be equal to S_{k-2}, i.e. the parallel reversal schedule for reversing l_{k-2} physical steps. Obviously, it would be advantageous to have appropriate parallel reversal schedules $\tilde{S}_{k-1}$ for reversing $\tilde{l}_{k-1} \equiv \bar{t}\, l_{k-2} - \hat{t} + 1$ at hand. Then S_k could be formed by the composition $S_k = S_{k-1} + \tilde{S}_{k-1}$. Furthermore, it is easy to prove by induction that $\tilde{l}_{k-1} > l_{k-1}$ for $k \geq 4$. It follows that the parallel reversal schedule $\tilde{S}_{k-1}$ needs more than $k-1$ resources for the reversal of $\tilde{l}_{k-1}$ physical steps. Therefore $S_k = S_{k-1} + \tilde{S}_{k-1}$ must be valid. Otherwise only $k-1$ resources can be used by $\tilde{S}_{k-1}$. Moreover, from the formula $S_k = S_{k-1} + \tilde{S}_{k-1}$ one obtains that $\tilde{S}_{k-1}$ needs no more than $k-1$ resources in the computational cycles after the vertex because at least one processor is needed by S_{k-1}. Hence one obtains for the resource profile $R(\tilde{S}_{k-1})$:

- $\exists\, j \in \{1, \ldots, \bar{t}\, \tilde{l}_{k-1}\} \;:\; R(\tilde{S}_{k-1}) \ni s^j = k$
- $R(\tilde{S}_{k-1}) \ni s^j \leq k-1 \quad \text{for} \quad \bar{t}\, \tilde{l}_{k-1} < j \leq (\bar{t}+1)\tilde{l}_{k-1} + \hat{t} - 1.$

For $\hat{t} < \bar{t}$ corresponding parallel reversal schedules $\tilde{S}_2$ and $\tilde{S}_3$ can be constructed according to the schedules depicted in Fig. 4.35 for $\bar{t} = 4$ and $\hat{t} = 2$. If one has $\hat{t} = \bar{t}$ similar parallel reversal schedules are used for $k \in \{3, 4\}$. Subsequently parallel reversal schedules $\tilde{S}_k$ are constructed recursively using $\tilde{S}_{k-1}$ and $\tilde{S}_{k-2}$. The corresponding procedure is depicted in Fig 4.36. An exact description how to construct the auxiliary feasible parallel reversal schedules $\tilde{S}_k$ is contained in the proof of Theorem B.1 for $\hat{t} < \bar{t}$ and in the proof of Theorem B.2 for $\hat{t} = \bar{t}$ (see Appendix B). Furthermore, the resulting resource profiles $R(\tilde{S}_k)$ are analyzed in detail. Because the construction is rather technical and does not provide further insights both theorems as well as their proofs are not presented here, but contained in Appendix B. Having the auxiliary parallel reversal schedules $\tilde{S}_k$ and their resource profiles at hand it is possible to prove the following assertion.

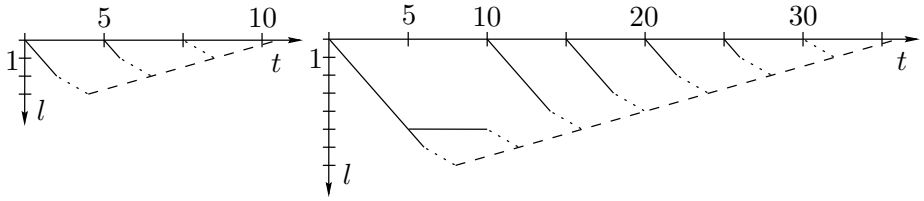


Figure 4.35: Parallel Reversal Schedules $\tilde{S}_2$ and $\tilde{S}_3$ for $\bar{t} = 4$ and $\hat{t} = 2$

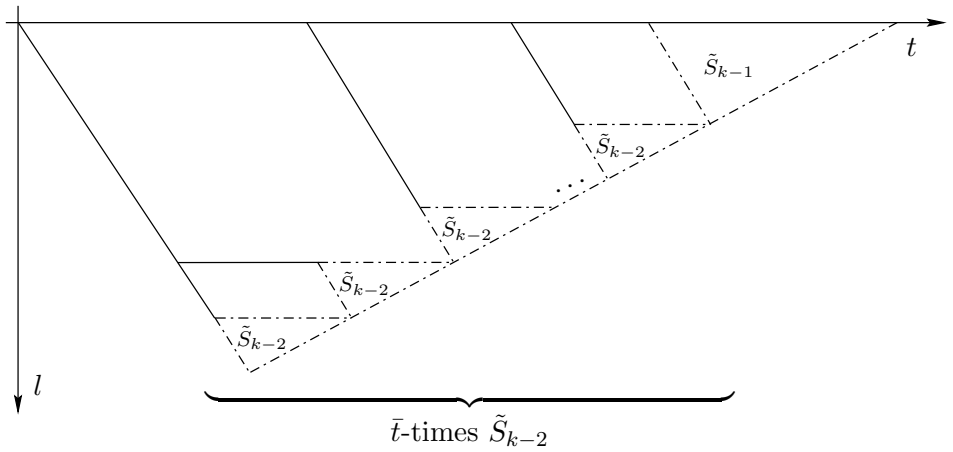


Figure 4.36: Construction of $\tilde{S}_k$ for $k > 3$

Theorem 4.10 (Reversal Schedules for $\bar{t} > 2$ and $\hat{t} \leq \bar{t}$).
Suppose the given uniform one-step evolution F determines the temporal complexities $\bar{t} > 2$ and $\hat{t} \leq \bar{t}$ to perform one reverse step and one recording step, respectively. Then there exist for $k > 4$ feasible parallel reversal schedules $S_k = S_{k-1} + \tilde{S}_{k-1}$ for the reversal of l_k physical steps, such that l_{k-1} and $\bar{t}\, l_{k-2} - \hat{t} + 1$ physical steps are reversed by S_{k-1} and $\tilde{S}_{k-1}$, respectively. For the resource profile $R(S_k)$, $k \geq 4$, follows

$$
\begin{array}{llll}
c_k^j = 0, & p_k^j = 1, & s_k^j = 1 & 1 \leq j \leq \bar{t} \\
 & p_k^j \leq \lceil \frac{i+1}{2} \rceil, & s_k^j \leq i & \bar{t}\, l_{i-1} < j \leq \bar{t}\, l_i,\ 2 \leq i \leq k \\
c_k^j = k-2, & p_k^j = 2, & s_k^j = k & j = \bar{t}\, l_k + 1 \\
c_k^j = k-2, & p_k^j = 1, & s_k^j = k-1 & 1 < j - \bar{t}\, l_k \leq l_3 + \hat{t} - 1 \\
c_k^j = k-i+1, & p_k^j = 1, & s_k^j = k-i+2 & \begin{cases} l_{i-1} < j - \bar{t}\, l_k - \hat{t} + 1 \leq l_i \\ 4 \leq i \leq k\ . \end{cases}
\end{array}
\tag{4.29}
$$

Proof. See Appendix B, Page 146. ∎

In order to prove Theorem 4.10 the resource profile $R(S_k)$ resulting from the composition of S_{k-1} and $\tilde{S}_{k-1}$ is analyzed. For that purpose the starting point is given by the feasible parallel reversal schedule S_3 the principle structure of which is depicted in Fig. 4.37 for $\bar{t} = 4$ and $\hat{t} = 2$. Subsequently the feasible parallel reversal schedule $S_k = S_{k-1} + \tilde{S}_{k-1}$, $k \geq 4$, illustrated in Fig. 4.38 is examined. A similar approach as for example in the proof of Theorem 4.5 yields

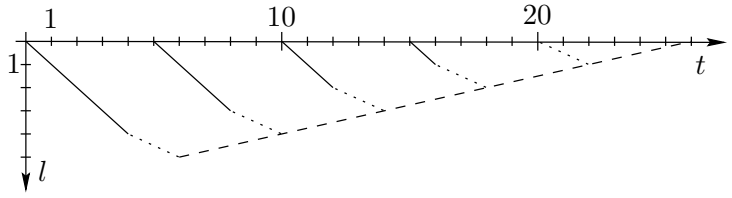


Figure 4.37: Parallel Reversal Schedule S_3 for $\bar{t} = 4$ and $\hat{t} = 2$

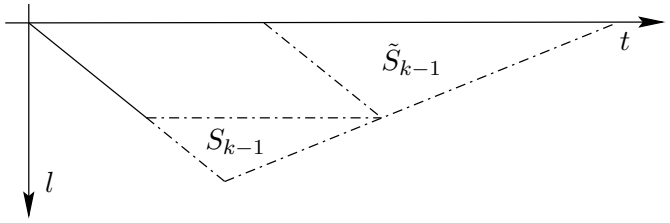


Figure 4.38: Parallel Reversal Schedule S_k for $k \geq 4$

that Equation (4.29) holds for the resource profile $R(S_k)$. Therefore the proof is not presented here, but contained in Appendix B. Because of Theorem 4.10 the assertion of Theorem 4.4 is valid for $(\hat{t}, \bar{t}) \in \mathbb{N} \times \mathbb{N}$ with $\hat{t} \leq \bar{t}$. Furthermore, one can conclude from the resource profile $R(S_k)$ that again the maximal number of resources is attained at least in the computational cycle $j = \bar{t}\, l_l + 1$ right after the vertex of S_k. Moreover, the upper bound of the resources required grows monotonously in a warm-up phase before the vertex and declines monotonously in a cool-down phase after the vertex. The same is true for the upper bound of the processors utilized by S_k.

At last the case $(\hat{t}, \bar{t}) \in \mathbb{N} \times \mathbb{N}$, $\hat{t} > \bar{t}$, will be considered. For the upper bound l_k one obtains according to Equation (4.3)

$$l_k \;=\; \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} .$$

For the construction of corresponding parallel reversal schedule S_k it is advantageous to have the following relation between the upper bound for the case $(\hat{t}, \bar{t}) \in \mathbb{N} \times \mathbb{N}$, $\hat{t} \leq \bar{t}$, and l_k for $(\hat{t}, \bar{t}) \in \mathbb{N} \times \mathbb{N}$, $\hat{t} > \bar{t}$, as defined in the last equation at hand.

Lemma 4.6.
Let $\hat{t}, \bar{t} \in \mathbb{N}$ with $\hat{t} > \bar{t} > 2$ be given and set $r \equiv \hat{t} - \lfloor (\hat{t} - 1)/\bar{t} \rfloor \bar{t} \in \{1, \ldots \bar{t}\}$. Define the sequences

$$l_k \;\equiv\; \begin{cases} k & \textit{if} \quad k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \textit{else} \end{cases} \qquad \textit{and}$$

$$l_{k,r} \;\equiv\; \begin{cases} k & \textit{if} \quad k \leq 2 \\ l_{k-1} + \bar{t}\, l_{k-2} - r + 1 & \textit{else} \end{cases}$$

with $k \in \mathbb{N}$. For $k \geq 2 + \hat{t}/\bar{t}$ follows that

$$l_k \;=\; l_{k-(\hat{t}-r)/\bar{t},r} + (\hat{t} - r)/\bar{t} \,. \tag{4.30}$$

Proof. In order to show Equation (4.30) consider $k = 2 + \lceil \hat{t}/\bar{t} \rceil$. According to the definition of l_k and $l_{k,r}$ this yields

$$\begin{aligned}
l_k &= k - 1 + \bar{t}(k-2) - \hat{t} + 1 = (\bar{t}+1)k - 2\,\bar{t} - \hat{t}, \\
l_{k-(\hat{t}-r)/\bar{t},r} &= l_{3,r} = 3 + \bar{t} - r, \\
l_{k+1} &= (\bar{t}+1)k - 2\,\bar{t} - \hat{t} + \bar{t}(k-1) - \hat{t} + 1 = 2\,\bar{t}\,k - 3\,\bar{t} - 2\,\hat{t} + k + 1, \\
l_{k-(\hat{t}-r)/\bar{t},r} &= l_{4,r} = 4 + 3\,\bar{t} - 2r\ .
\end{aligned}$$

Therefore one has that

$$\begin{aligned}
l_k &= 2 + \lceil \hat{t}/\bar{t}\,\rceil + \bar{t} - r = l_{k-(\hat{t}-r)/\bar{t},r} + (\hat{t}-r)/\bar{t} \quad \text{and} \\
l_{k+1} &= 3\,\bar{t} + 3 + \lceil \hat{t}/\bar{t}\,\rceil - 2r = l_{k+1-(\hat{t}-r)/\bar{t},r} + (\hat{t}-r)/\bar{t}\ .
\end{aligned}$$

Now assume that Equation (4.30) is valid for $2 + \lceil \hat{t}/\bar{t}\,\rceil \leq j \leq k-1$ with $k > 3 + \lceil \hat{t}/\bar{t}\,\rceil$. Then one obtains

$$\begin{aligned}
l_k &= l_{k-1} + \bar{t}\,l_{k-2} - \hat{t} + 1 \\
&= l_{k-1-(\hat{t}-r)/\bar{t},r} + (\hat{t}-r)/\bar{t} + \bar{t}\left(l_{k-2-(\hat{t}-r)/\bar{t},r} + (\hat{t}-r)/\bar{t}\right) - \hat{t} + 1 \\
&= l_{k-1-(\hat{t}-r)/\bar{t},r} + \bar{t}\,l_{k-2-(\hat{t}-r)/\bar{t},r} - r + 1 + (\hat{t}-r)/\bar{t} \\
&= l_{k-(\hat{t}-r)/\bar{t},r} + (\hat{t}-r)/\bar{t}\ .
\end{aligned}$$

Hence Equation (4.30) holds for all $k \geq 2 + \hat{t}/\bar{t}$. ∎

Equation (4.30) is exploited in the proof of the following theorem in order to construct the desired parallel reversal schedules S_k.

Theorem 4.11 (Reversal Schedules for $\bar{t} > 2$ and $\hat{t} > \bar{t}$).
Suppose the given uniform one-step evolution F determines the temporal complexities $\hat{t} > \bar{t} > 2$ to perform one recording step and one reverse step, respectively. Let k resources be available each of which has the property of processor-checkpoint convertibility. It is possible to construct feasible parallel schedules S_k for the reversal of l_k physical steps which need no more than k resources at any time on the base of the feasible parallel schedules developed in Theorem 4.10. The number of processors needed by S_k does not exceed

$$p_k^b \equiv \begin{cases} k & \text{if } 1 \leq k < 2 + \hat{t}/\bar{t} \\ \lceil (k+1)/2 \rceil + \left\lceil \frac{1}{2} \lfloor (\hat{t}-1)/\bar{t} \rfloor \right\rceil & \text{else} \end{cases} .$$

Proof. See Appendix B, Page 149. ∎

In the proof of Theorem 4.11 the integer value $r = \hat{t} - \lfloor (\hat{t}-1)/\bar{t} \rfloor \bar{t}$ is determined. The desired S_k to reverse l_k physical steps is built on the parallel reversal schedule $S_{k,r}$ for the pair $(r, \bar{t}\,)$ as constructed in Theorem 4.10. For that purpose the recording steps of $S_{k,r}$ are modified such that the given value of $\hat{t} > \bar{t}$ is taken into account. Subsequently the reversal of $(\hat{t}-r)/\bar{t}$ additional physical steps is attached at the vertex of $S_{k,r}$. As can be seen the argument is similar to the one used in the proof of Theorem 4.6. Therefore the proof of Theorem 4.11 is not presented here, but contained in Appendix B.

Because of Theorem 4.11 the assertion of Theorem 4.4 holds also for the case considered at last. Therefore it has been proven for all natural number pairs $(\hat{t}, \bar{t}\,) \in \mathbb{N} \times \mathbb{N}$.

4.4 Conclusions

Let an one-step evolution F be given that determines the temporal complexities $\bar{t} \in \mathbb{N}$ and $\hat{t} \in \mathbb{N}$ to perform a reverse step and a recording step, respectively. Throughout this chapter it was assumed that each resource that is available has the property of processor-checkpoint convertibility. Having this simplification at hand it was proven in Section 4.2 that the maximal number l_k of physical steps that can be reversed with up to k resources at any time is limited above in the following way

$$l_k \;\leq\; \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} .$$

Section 4.3 contains the construction of feasible parallel reversal schedules that apply up to k resources at any time and reverse exactly k physical steps if one has $k < 2 + \hat{t}/\bar{t}$ and $l_k \equiv l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1$ physical steps otherwise. Therefore it was proven that for the maximal number l_k of physical steps that can be reversed with up to k processors and checkpoints at any time the equality

$$l_k \;=\; \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/\bar{t} \\ l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 & \text{else} \end{cases}$$

holds. Furthermore, the construction of the parallel reversal schedules S_k yields an upper bound of the fraction of processors among the k processors and checkpoints. It was shown that no more than

$$p_k^b \;\equiv\; \begin{cases} k & \text{if} \quad k < 2 + \hat{t}/\bar{t} \\ \lceil (k+1)/2 \rceil + \left\lceil \frac{1}{2} \lfloor (\hat{t} - 1)/\bar{t} \rfloor \right\rceil & \text{else} \end{cases}$$

processors are required in each computational cycle. So far, for general temporal complexities $\bar{t}, \hat{t} \in \mathbb{N}$ there is no idea how to prove that this fraction of processors is really needed or how to reduce p_k^b further.

Figure 4.39 serves to get a better impression of the parallel reversal schedules developed in the last section. Here for $\bar{t} = \hat{t} = 1$ the parallel reversal schedule S_7 as well as the needed checkpoints c_7^j, the needed processors p_7^j and the sum s_7^j of both are shown. As can be seen there is a warm-up and a cool-down phase, where the number of processors and checkpoints required increases and decreases, respectively. Furthermore, the maximal number of processors and checkpoints, namely 7, is needed during the recording step $F_{l_k - 1}$. This fact is true for all parallel reversal schedules that reverse the maximal number of physical steps with a given number of processors and checkpoints. Otherwise it would be possible to reverse at least one more physical step.

To illustrate the achieved results further assume that at most 16 processors are available and at most 16 checkpoints can be accommodated. This is a

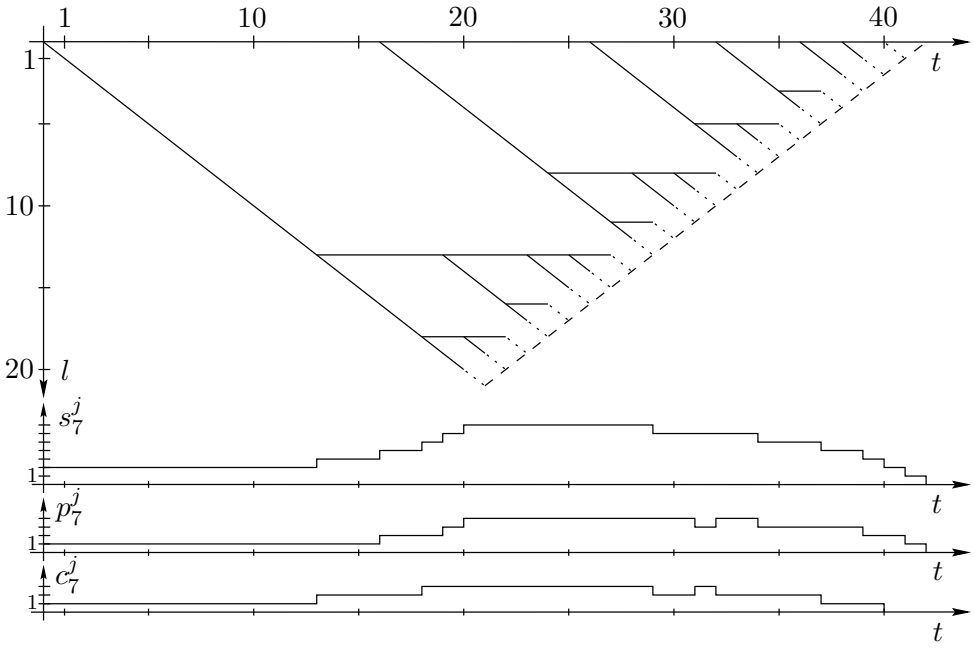


Figure 4.39: Parallel Reversal Schedule S_7 for $\bar{t} = \hat{t} = 1$

quite realistic situation for existing multi-processor machines. Then for several combinations of $\bar{t}$ and $\hat{t}$ the corresponding value of l_k, i.e. the maximal number of physical steps that can be reversed with the parallel reversal schedule S_k developed in this chapter, are given by Table 4.1. Because of the size of l_k,

$\bar{t} \backslash \hat{t}$	1	2	3
1	2 178 309	514 230	317 813
2	1 073 741 824	715 827 883	134 217 729
3	65 523 692 457	51 830 637 705	36 137 582 953

Table 4.1: Maximal Number l_k of Physical Steps that can be reversed

it should be possible to reverse every one-step evolution of interest for these temporal complexities.

Now the behavior of l_k is analyzed in more detail. For $\hat{t} = \bar{t} = 1$ and $k > 0$ follows that l_k equals the $(k-1)$th Fibonacci number. Therefore one finds that (see e.g. [Knu73])

$$l_k \sim \left(\frac{1}{2}(1+\sqrt{5})\right)^{k-1} . \tag{4.31}$$

This relation yields that the number of physical steps that can be reversed with k resources grows exponentially in k. For the corresponding reversals $\lceil (k+1)/2 \rceil$ processors suffice as shown in Section 4.3.

If the given one-step evolution determines another combination of $\bar{t}$ and $\hat{t}$ then a slightly different formulation of the recursion provides more insight. One

obtains for $k \geq 2 + \hat{t}/\bar{t}$

$$l_k = l_{k-1} + \bar{t}\, l_{k-2} - \hat{t} + 1 = l_{k-1} + \bar{t}\, l_{k-2} + \alpha\, \bar{t}$$

with $\alpha = (1 - \hat{t})/\bar{t}$. This yields with $\bar{l}_k \equiv l_k + \alpha$

$$\bar{l}_k = \bar{l}_{k-1} + \bar{t}\, \bar{l}_{k-2} \ ,$$

which equals the recursion formula for the generalized Fibonacci sequence (see e.g. [HP98]). Hence, one can derive a behavior of the corresponding l_k that is similar to the right-hand side of Equation (4.31). Here, only the base differs. In detail one has

$$l_k \sim \left(\frac{1}{2}(1 + \sqrt{1 + 4\bar{t}}) \right)^{k-1} \ .$$

Therefore also for general $\bar{t}$ and $\hat{t}$ one has an exponential growth of l_k in k.

Moreover, it is possible to draw the following interesting connection between serial and parallel reversal schedules. Assume that the given one-step evolution F combines l physical steps and determines the temporal complexities $\bar{t} = 1$ and $\hat{t} = 1$ to perform one reverse step and one recording step with one processor, respectively. Let P processors be available. Furthermore, suppose that the times required to perform one physical step, one reverse step, and one recording step speedup perfectly with respect to the number of processors that are applied to perform them. Thus, if m denote the number of processors that can be used then the time needed to perform either a physical step, a reverse step, or a recording step is given by $1/m$. For the number p of processors required by the parallel reversal schedules S_k developed in Section 4.3 to reverse l physical steps follows

$$p \quad \approx \quad 0.5 \log_{(1+\sqrt{5})/2} l$$

The time t_M needed for the complete reversal of l physical steps is determined by $t_M = 2\, l/m$ taking the speed-up of the physical steps, reverse steps, and recording steps into account. With $m = P/p$ one obtains $t_M = 2\, lp/P$. Assume for the serial reversal of F that $r = c$, where r is defined as in Chapter 3. I.e. a similar growth in the temporal and spatial complexity of the serial reversal is accepted. Then $r\, l$ can be seen as integer approximation of the cost to reverse the given l physical steps using optimal serial reversal schedules. Using the processor-checkpoint convertibility one can assume $r = c = p$. Hence, the time required to reverse l physical steps with the serial reversal schedule equals lp/P because of the perfect speed-up. In this special case the ratio between the time needed for the reversal using a optimal serial reversal schedule and the time needed for the reversal using a optimal parallel schedule is given by 2. For the serial reversal schedule P processors are active throughout the reversal process. Using the parallel reversal schedule it seems that only $P/2$ processors work on an average in time. These facts can be seen as an explanation for the asymptotic factor 2. Furthermore, the serial reversal schedule that takes the speed-up into account can be used even if the step costs vary slightly. However,

it depends strongly on the perfect speed-up. The parallel reversal schedule needs uniform-step costs but can handle a speed-up that is not as regular as assumed here. Obviously the unrestricted speed-up of the run time needed to perform a physical step, a recording step or a reverse step is not realistic. Nevertheless, it might be useful to utilize the processors that are actually not needed for the reversal process to speed up the evaluation of the physical steps. For example, this could be done in the warm-up or cool-down phase. The development of corresponding parallel reversal schedules will be subject of future work.

Usually, the one-step evolution under consideration does not combine a number l of physical steps that equals l_k for some $k \in \mathbb{N}$. Nevertheless, it is possible to determine one unique $k \in \mathbb{N}$ with $l_{k-1} < l \leq l_k$. Then it is obvious that exactly k resources are needed to reverse the given l physical steps. Here, again in each computational cycle a resource may be used either as processor or as checkpoint. Furthermore, the corresponding parallel reversal schedule S_k developed in Section 4.3 can be applied with slight modifications to reverse the given number l of physical steps. To that end more or less only the actions of S_k are performed that act on the states $l_k - l, \ldots, l_k$ because they form a complete reversal schedule for l physical steps.

Chapter 5

Conclusions and Outlook

Often one may need to reverse the evaluation of a given function F. The key difficulty is given by the enormous amount of memory required for the reversal of any evaluation program that has a reasonable size if the basic approach is used. If F can be split into a sequence of physical steps then F forms an evolutionary system. Different types of evolutionary systems are considered. The goal of this thesis was to present optimal strategies to reverse these evolutionary systems. The proposed reversal schedules for single- and multi-processor machines reduce the spatial complexity for reversing F drastically.

5.1 Serial Reversal Schedules

Suppose the reversal of F has to be performed on a single-processor machine. The case where F combines physical steps of the same temporal complexity and each state depends only on the previous one has been completely covered in the literature. Only little was known about the reversal of evolutionary systems with non-uniform step costs.

This thesis presents new results with respect to optimal serial reversal schedules for multi-step evolutions with uniform step costs. Here each state depends on more than one previous state and the cost to go from the previous state to the next one is always the same. The derived serial reversal schedules minimize the number of physical steps that have to be performed during the reversal. Furthermore, they cause the minimal number of checkpoint writings. The routine `revolve.c` implementing the developed optimal serial reversal schedules was coded. As numerical application it was used to approximate the solution of an adjoint differential equation. Three different multi-step recurrences were considered. The achieved results were reported and interpreted.

In the case of one-step recurrences and non-uniform step costs a search algorithm has to be used to find one optimal serial reversal schedule. Based on the technique of dynamic programming an obvious search algorithm has a temporal complexity that is cubic in the number of physical steps to be reversed. In this thesis an important property of the checkpoint writings is proven, namely the "monotonic partition". This property can be applied to reduce the complexity of the search algorithm. For all considered examples the

time required to find an optimal serial reversal schedule is drastically reduced, apparently by one power of l. It may be conjectured from the measured run times that it is possible to achieve a temporal complexity of the search algorithm that is quadratic in the number of physical steps to be reversed. Nevertheless two tasks are left for future work. On one hand it would be advantageous to have a proof of the reduction to a quadratic complexity or even further. For monotonously increasing or decreasing step costs it should not be too difficult to obtain a corresponding result. For arbitrary step costs so far there is no idea how the proof could work. On the other hand, usually one does not know the exact distribution of the step costs a priori as assumed in this thesis. Therefore it is desirable to have a procedure at hand, which performs the checkpoint writings during the first forward sweep from the initial state to the penultimate state according to a predefined strategy and stores the step costs observed in an array. This array can be used during the following forward sweeps to manage the checkpoint writings such that the time needed for the reversal is minimized. The steering process could be based on the improved search algorithm presented in Chapter 3, but one has to think about the strategy applied to perform the checkpoint writings in the first forward sweep.

5.2 Parallel Reversal Schedules

So far, one knew only little about reversal strategies on multi-processor machines. Therefore the development of optimal parallel reversal schedules is right at the beginning. This thesis answers for uniform one-step evolutions the question how many physical steps can be reversed maximally with a given number of resources. Here it is supposed that each resource has the property of processor-checkpoint convertibility, i.e. can be used either as processor or as checkpoint in each computational cycle. The corresponding parallel reversal schedules are derived in Chapter 4. These parallel reversal schedules also can be used easily for evolutionary systems that use multi-step recurrences in the following way. Suppose that a q-step recurrence is applied. One can group always q physical steps to one mega step. Then the reversal of the modified evolutionary system can be performed by the parallel reversal schedules developed in the last chapter. Obviously this is only a sub-optimal solution if additional reverse steps can be performed during the release of a checkpoint as considered in Chapter 3.

It is planned to code the parallel reversal schedules of Chapter 4 using either MPI or OpenMP. With respect to the implementation the following remarks can be made. First one has to note that the communication between the processors is largely determined a priori. In principle only checkpoints need to be exchanged. Furthermore if there are more checkpoints than are needed by the particular parallel reversal schedule, then they can be used to manage possible delays of the processors. Moreover assume that the given evolutionary system F is coded using already a parallel mechanism. Suppose that m processors are required to evaluate F. Let p be the number of processors that are needed by a feasible parallel reversal schedule S to reverse the evolutionary system F. Then the number of processors required for the reversal with the schedule S is

given by the product of m and p. A further task for the future should be the development of suitable recover strategies if there are delays or break downs during the calculation of reversal processes.

Besides these more practical aspects there are several theoretical questions to think about. Throughout Chapter 4 it was assumed that the temporal complexities $\bar{t}$ and $\hat{t}$ determined by the given evolutionary system are natural numbers. Obviously, this assumptions forms a restriction of the allowed evolutionary systems that is not negligible. Therefore in the future one has to examine how the theory for the reversal of evolutionary systems on multi-processor machines can be extended to more general situations. Furthermore, so far only uniform step costs are considered. Optimal reversal schedules for multi-processors machines and non-uniform step costs are an open research topic, where so far no results are known.

5.3 Summary

In order to summarize the above it appears useful to work on the development of optimal reversal strategies for program evolutions because the enormous amount of memory required to reverse a program execution is still a problem.

For a wide range of applications, namely evolutionary systems that have uniform step costs and employ one- or the multi-step recurrences considered in this thesis, software tools are available to reverse the corresponding evaluation on serial machines. Furthermore now an improved search algorithm is coded to find optimal serial reversal schedules in the case of non-uniform step costs. This should be a good base to reverse program evolutions that describe evolutionary systems on serial machines.

Moreover now optimal schedules are available for the reversal of many evolutionary systems on multi-processor machines. On one hand the results achieved in this thesis form a good foundation to implement a corresponding software tool that manages the reversal process on multi-processor machines appropriately. On the other hand now a first base for further theoretical and practical results with respect to more general evolutionary systems is built.

Appendix A

Source of the Coded Algorithms

A.1 `revolve.c`: Multi-step Recurrences and Uniform Step Costs

The procedure `revolve.c` below is self-contained. It can be used to control the reversal process on a serial machines for a given q-step evolution determining the parameter b as described in Section 3.3.2 of Chapter 3. For that purpose the technique of reverse communication has to be utilized.

```
#include <stdio.h>
#include <stdlib.h>
#define min(x,y) ((y<x) ? y : x)

enum action { advance, takeshot, restore, firsturn, youturn,
              terminate, error, reduce_checkpoint};

enum action revolve(int* check,int* capo,int* fine,int snaps,
                    int q,int b,int* perform,int* info)
 {
  static int ch[checkup], m, range_b, range_q, reduce,
             reps, turn;
  int ds, mh, num, oldcapo;
  int bino1, bino2, bino3, bino4, bino5, bino6, bino7;
    /* (*capo,*fine) = time range under consideration */
    /* ch[j] = number of the state stored in checkpoint j */

  if (*check < -1 || *capo > *fine || b >= q)
    return error;
  if ((*check == -1) && (*capo < *fine))
   {
    if (*check == -1)
     {
```

```
     turn = 0;     /* initialization of turn counter */
     reduce = 0;   /* do we have to go b steps back? */
     m = 0;
    }
   *ch = *capo-1;
  }
 if (*fine-*capo == 0)
  {   /* reduce capo to previous checkpoint, unless done  */
   if (*check == -1 || *capo==*ch)
    {
     *check -= 1;
     return terminate;
    }
   else
    {
     if ((reduce == 1) && (b > 0))
      {
       reduce = 0;
       *fine = *fine - b;
       *capo = *fine;
       return reduce_checkpoint;
      }
     else
      {
       *capo = ch[*check];
       m = 0;
       return restore;
      }
    }
  }
 else
  {
   if (*fine-*capo <= q)
    {     /* (possibly first) combined forward/reverse step */
     *perform = *fine-*capo;
     *fine = *capo;
     if (*check >= 0 && ch[*check] == *capo)
      {
       *check -= 1;
       reduce = 1;
      }
     if (turn == 0)
      {
       turn = 1;
       return firsturn;
      }
     else
```

```
      return youturn;
   }
  else
   {
    if (*check == -1 || ch[*check] != *capo)
     {
      *check += 1 ;
      ch[*check] = *capo;
      return takeshot;
     }
    else
     {
      oldcapo = *capo;
      ds = snaps - *check;
      reps = 0;
      range_b = 1;
      range_q = 1;
      while(q * range_q + b * (range_b - 1) < *fine - *capo)
       {
        reps += 1;
        range_q = range_q*(reps + ds)/reps;
        range_b = range_b*(reps + ds - 1)/reps;
       }
      bino1 = range_q*reps/(ds+reps);
      bino2 = (ds > 1) ? bino1*ds/(ds+reps-1) : 1;
      if (ds == 1)
        bino3 = 0;
      else
        bino3 = (ds > 2) ? bino2*(ds-1)/(ds+reps-2) : 1;
      bino4 = bino2*(reps-1)/ds;
      bino6 = range_q*ds/(ds+reps);
      if (reps == 1)
        bino7=0;
      else
        bino7 = (reps > 2) ? bino2*(reps-1)/(ds+reps-2): 1;
      if (*fine-*capo <= q*(bino1+bino3) + b*(bino2-1))
        *capo = *capo+q*bino4+b*bino7;
      else
       {
        if (ds == 1)
          *capo = *fine-q;
        else
         {
          if (*fine-*capo <= q*(bino1 + bino2) + b*(bino2-1))
            *capo = *fine-q*(bino2+bino3)-b*(bino3-1);
          else
           {
```

```
              if (turn == 0)
               {
                if (m == 0)
                  m = min(range_q,*fine);
                while(*fine-*capo <= (m-1)*q+((m-1)-bino1-1)*b)
                  m -= 1;
               }
              else
               {
                m = bino1 + bino2 + 1;
                while(*fine-*capo > m*q+(m-bino1-1)*b)
                  m += 1;
               }
              if (ds < 3)
                bino5 = 0;
              else
                bino5 = (ds > 3) ? bino3*(ds-2)/reps : 1;
              if (m > bino1 + bino2 + bino5)
                mh = (m-bino6);
              else
                mh = (bino4 + bino7+1);
              *capo = *fine-q*(m-mh)-b*(m-mh-bino2-1);
            }
          }
        if ((*capo-oldcapo < q) && (*fine-*capo > q))
          *capo = oldcapo+q;
        if (*fine-*capo < q)
          *capo = *fine-q;
        return advance;
       }
     }
   }
}
```

If $c = 3$ checkpoints are available a procedure similar to the code list at Page 44 causes for $l = 20$ and the parameters $q = 2$ and $b = 1$ (e.g. leap-frog method) the output:

```
ENTER:   STEPS, CHECKS, Q, B, INFO
 20   3   2   1   3

prediction of needed forward steps:       28 =>
slowdown factor:   1.4000

takeshot at           0
advance to            6
takeshot at           6
```

```
advance to           14
takeshot at          14
advance to           18
firsturn at          18   with    2 steps
restore at           14
advance to           16
youturn at           16   with    2 steps
restore at           14
youturn at           14   with    2 steps
reduce checkpoint at 14
restore at            6
advance to            9
takeshot at           9
advance to           11
youturn at           11   with    2 steps
restore at            9
youturn at            9   with    2 steps
reduce checkpoint at  9
restore at            6
youturn at            6   with    2 steps
reduce checkpoint at  6
restore at            0
advance to            2
takeshot at           2
advance to            3
youturn at            3   with    2 steps
restore at            2
youturn at            3   with    2 steps
youturn at            2   with    1 steps
reduce checkpoint at  2
restore at            0
youturn at            0   with    1 steps

advances:    28
takeshots:    5
commands:    35
```

A.2 sched.c: One-step Recurrences and Non-uniform Step Costs

The complete source of the improved search algorithm for an optimal serial reversal schedule as described in Section 3.4 of Chapter 3 is listed below. Apart from the procedure `initweights(..)` it is self-contained. The routine `initweights(..)` initializes the array `w` of step costs. Therefore it has to be written by the user in correspondence to the evolutionary system under consideration.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

double** t;
double* w;
double* sum;
int** part;

main()
{
int c,d,h,i,l,m,lb_m,ub_m;

double slope,totalcost,tenp;
printf(" checks, steps, slope \n"); */
scanf("%d %d %lf",&c,&l,&slope);
t = (double**)malloc((c+1)*sizeof(double*));
part = (int**)malloc((c+1)*sizeof(int*));
w = (double*)malloc(l*sizeof(double));
sum = (double*)malloc(l*sizeof(double));

for(h=1;h<=c;h++)
{ t[h] = (double*)calloc(l*(l+1)/2,sizeof(double));
  part[h] = (int*)calloc(l*(l+1)/2,sizeof(int));
}

initweights(w,sum,l,slope);
for(s=1;s<=c;s++)
{ for(d=2;d<=l;d++)
   for(i=0;i<=l-d;i++)
   { h=i+d;
     if (d <= s+1)
     { if (i > 0)
        t[s][i+h*(h-1)/2-1]=sum[h-2]-sum[i-1];
       else
        t[s][i+h*(h-1)/2-1]=sum[h-2];
       part[s][i+h*(h-1)/2-1]=i+1;
     }
     else
     { if (s == 1)
       { if (d == 2)
          t[s][i+h*(h-1)/2-1]=2*w[i]+w[i+1];
         else
          if (i == 0)
           t[s][i+h*(h-1)/2-1]=t[s][i+(h-1)*(h-2)/2-1]+sum[h-2];
          else
           t[s][i+h*(h-1)/2-1]=t[s][i+(h-1)*(h-2)/2-1]
```

```
                                +sum[h-2]-sum[i-1];
        }
        else
        { lb_m = part[s][i+(h-1)*(h-2)/2-1];
          ub_m = part[s][i+1+h*(h-1)/2-1];
          part[s][i+h*(h-1)/2-1] = 0;
          t[s][i+h*(h-1)/2-1] = -1;
          for(m=lb_m;m<=ub_m;m++)
          { if (i > 0)
             tenp = sum[m-1]-sum[i-1];
            else
             tenp = sum[m-1];
            tenp += t[s-1][m+h*(h-1)/2-1] + t[s][i+m*(m-1)/2-1];
            if(tenp<t[s][i+h*(h-1)/2-1] || t[s][i+h*(h-1)/2-1]<0)
            { t[s][i+h*(h-1)/2-1] = tenp;
              part[s][i+h*(h-1)/2-1] = m;
            }
          }
        }
      }
    }
}

printf("\n Minimal Reversal Cost: %10.2f \n",t[c][l*(l-1)/2]);
schedprint(c,0,l);
}
```

Appendix B

Construction of Feasible Parallel Reversal Schedules

B.1 Construction of Parallel Reversal Schedules for $\bar{t} = 2$ and $\hat{t} > 1$

This section contains the proofs of the Theorems 4.8 and 4.9. In the first proof the desired feasible parallel reversal schedules S_k for the temporal complexities $(\hat{t}, \bar{t}\,) = (2, 2)$ are constructed. The second one describes how to develop the desired feasible parallel reversal schedules S_k if the given uniform one-step evolution determines the temporal complexities $\bar{t} = 2$ and $\hat{t} > 2$.

Proof of Theorem 4.8, Page 91:
For $k \leq 3$ one has $l_k = k$. It is obvious how to construct the corresponding parallel reversal schedules S_k such that they need no more than $\lceil (k+1)/2 \rceil$ processors. In order to prove the existence of appropriate parallel reversal schedules for an arbitrary $k > 3$ consider the parallel reversal schedules S_3 and S_4 as shown in Fig. 4.34. It is easy to see that both schedules need no more than 3 and 4 resources in any computational cycle, respectively. Furthermore, one finds that the number of processors used does not exceed $\lceil (3+1)/2 \rceil$ and $\lceil (4+1)/2 \rceil$, respectively. The parallel reversal schedules S_k for $k \geq 5$ are constructed according to the procedure illustrated by Fig. 4.33, where $\bar{S}_{k-2}$ denotes the reduced parallel reversal schedule S_{k-2} to reverse $l_{k-2} - 1$ physical steps. In order to create $\bar{S}_{k-2}$ the parallel reversal schedule S_{k-2} is shortened by the reversal of the last physical step $F_{l_{k-2}-1}$.

Using an induction on k it will be shown in this proof that Equation (4.27) holds for the resource profile $R(S_k)$, $k > 4$, if S_k is recursively constructed according to Fig. 4.33.

Considering the resulting parallel reversal schedule S_5 (see Fig. B.1) it follows that Equation (4.27) holds for the resource profile $R(S_5)$. The reversal schedule S_6 constructed according to Fig. 4.33 as well as the resulting resource profile $R(S_6)$ are illustrated by Fig. B.2. One finds that Equation (4.27) is also valid for $R(S_6)$.

Now it will be proven that Equation (4.27) holds for $R(S_k)$, $k > 6$, if S_k

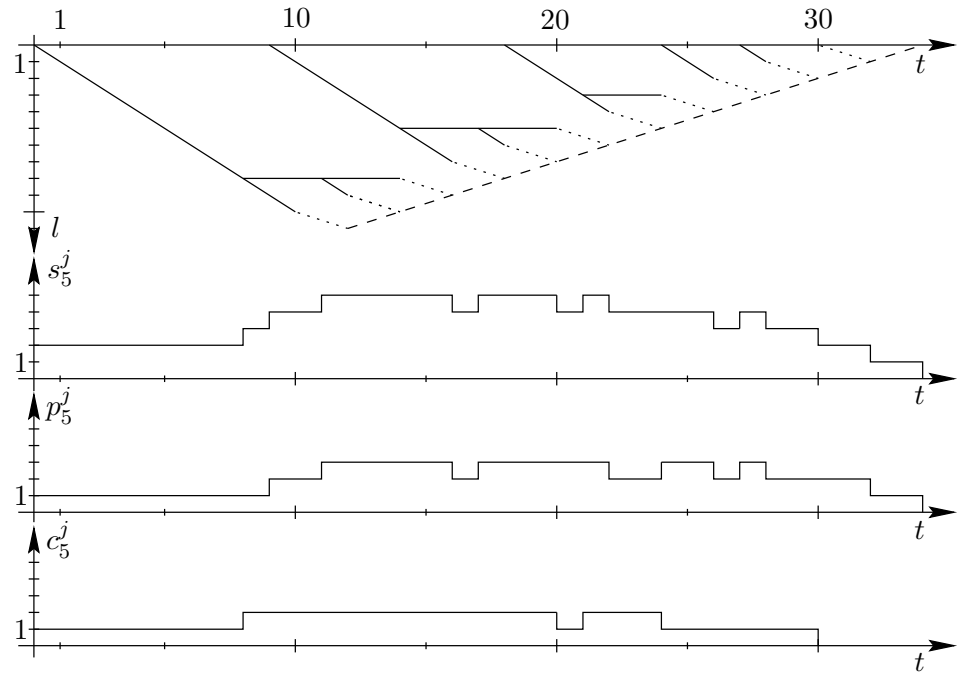


Figure B.1: Parallel Reversal Schedule S_5 for $\hat{t} = 2$ and $\bar{t} = 2$

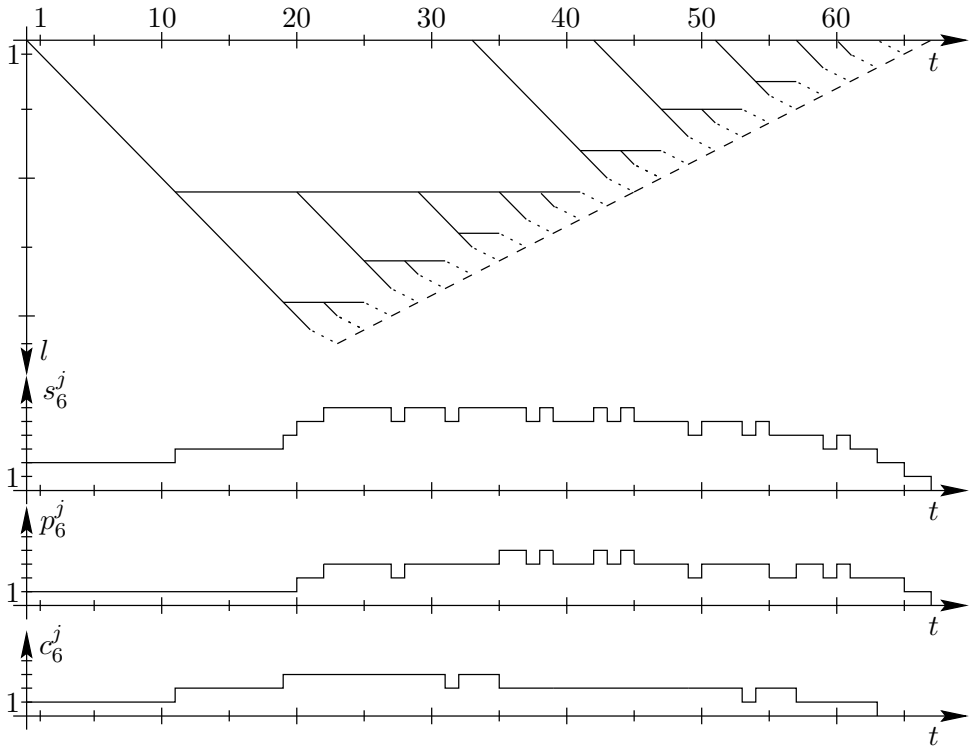


Figure B.2: Parallel Reversal Schedule S_6 for $\hat{t} = 2$ and $\bar{t} = 2$

is constructed in accordance with Fig. 4.33. First, suppose that k is odd. The alternative will be considered later.

If k is odd S_k is constructed on the base of S_{k-2} and $\bar{S}_{k-2}$, where $\bar{S}_{k-2}$ results from a shortening of S_{k-2} as as depicted in Fig. B.3 for $k-2=5$. The lines drawn in black determine the new parallel reversal schedule $\bar{S}_{k-2}$. It will

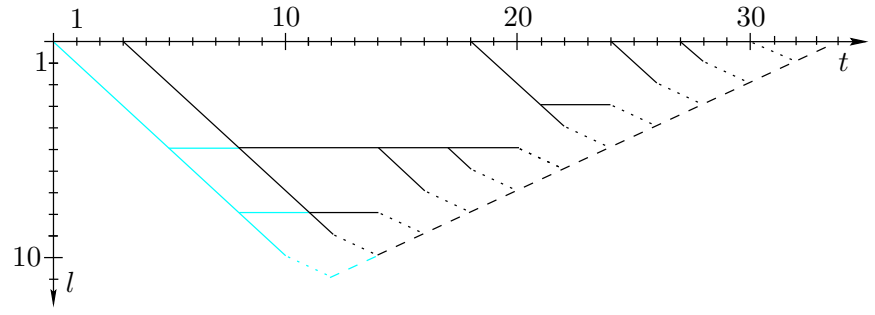


Figure B.3: Reversal Schedule $\bar{S}_5$

be shown using once more an induction on k that for $\bar{S}_{k-2}$ the resource profile $R(\bar{S}_{k-2})$ fulfils

$$
\begin{array}{llll}
\bar{c}^j_{k-2}=0, & \bar{p}^j_{k-2}=1, & \bar{s}^j_{k-2}=1 & j=1,2 \\
 & \bar{p}^j_{k-2}\le\lceil\frac{i+1}{2}\rceil, & \bar{s}^j_{k-2}\le i & \begin{cases} l_i<j\le l_{i+1}+1,\ i\ \text{even} \\ l_i+1<j\le l_{i+1},\ i\ \text{odd} \\ i=2,\dots,k-3 \end{cases} \\
 & \bar{p}^j_{k-2}\le\frac{k-1}{2}, & \bar{s}^j_{k-2}\le k-2 & l_{k-2}+1<j\le l_{k-1}-2 \\
\bar{c}^j_{k-2}=\frac{k-3}{2}, & \bar{p}^j_{k-2}=\frac{k-3}{2}, & \bar{s}^j_{k-2}=k-3 & l_{k-1}-1\le j\le l_{k-1}+1 \\
\bar{c}^j_{k-2}=\frac{k-5}{2}, & \bar{p}^j_{k-2}=\frac{k-3}{2}, & \bar{s}^j_{k-2}=k-4 & j=l_{k-1}+2, l_{k-1}+3 \\
\bar{c}^j_{k-2}=\frac{k-i}{2}, & \bar{p}^j_{k-2}=\frac{k-i}{2}, & \bar{s}^j_{k-2}=k-i & l_{i-2}<j-l_{k-1}\le l_i-2 \\
\bar{c}^j_{k-2}=\frac{k-i-2}{2}, & \bar{p}^j_{k-2}=\frac{k-i}{2}, & \bar{s}^j_{k-2}=k-i-1 & l_i-2<j-l_{k-1}\le l_i \\
 & & & i=5,7,\dots,k-4 \\
\bar{c}^j_{k-2}=1, & \bar{p}^j_{k-2}=1, & \bar{s}^j_{k-2}=2 & l_{k-1}+l_{k-4}<j\le 3\,l_{k-2}-2\,.
\end{array}
\tag{B.1}
$$

Equation (B.1) is valid for the resource profile $R(\bar{S}_5) = R(\bar{S}_{7-2})$ as can be seen from Figs. B.3 and B.4. This fact will be exploited to construct for odd $k > 5$ parallel reversal schedules S_k such that Equation (4.27) holds for the resource profiles $R(S_k)$. Subsequently it will be shown that Equation (B.1) is valid for the resource profiles $R(\bar{S}_k)$ of the shortened parallel reversal schedules $\bar{S}_k$.

For constructing the feasible parallel schedule to reverse l_k physical steps S_k takes the first l_{k-2} computational cycles over from S_{k-2} without any changes. Hence one has for $j = 1$ and $j = 2$

$$
c^j_k = c^j_{k-2} = 0, \quad p^j_k = p^j_{k-2} = 1, \quad s^j_k = s^j_{k-2} = 1 \tag{B.2}
$$

as well as for $i = 2, \dots, k-2$

$$
p^j_k = p^j_{k-2} \le \lceil\tfrac{i+1}{2}\rceil, \quad s^j_k = s^j_{k-2} \le i \quad \begin{cases} l_i < j \le l_{i+1}+1,\ i\ \text{even} \\ l_i+1 < j \le l_{i+1},\ i\ \text{odd} \end{cases} \tag{B.3}
$$

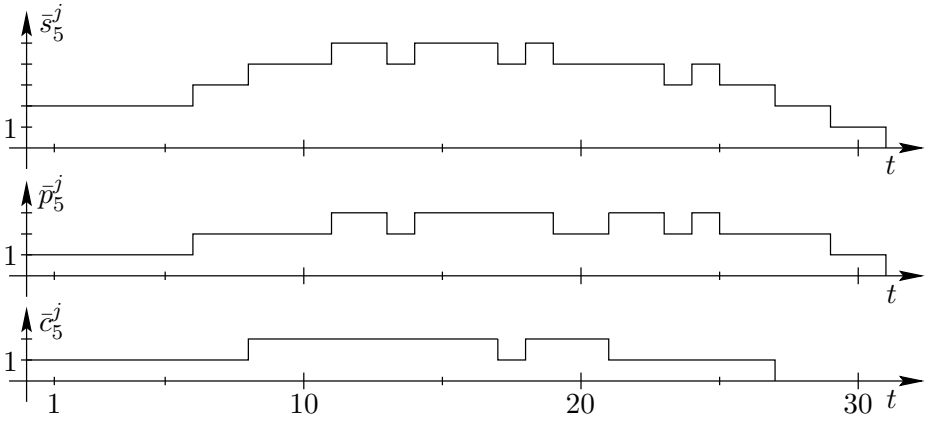


Figure B.4: Resource Profile $R(\bar{S}_5)$

and the vertex of the first parallel reversal schedule S_{k-2} is reached. The parallel reversal schedule $\bar{S}_{k-2}$ starts. This yields for $j = l_{k-1} + 1 = 2\,l_{k-2} + 1$ and $j = 2\,l_{k-2} + 2$

$$p_k^j = p_{k-2}^j + \bar{p}_{k-2}^{j-2l_{k-2}} \leq \left\lceil \frac{k-2}{2} \right\rceil + 1 = \left\lceil \frac{k}{2} \right\rceil, \quad s_k^j = s_{k-2}^j + \bar{s}_{k-2}^{j-2l_{k-2}} \leq k-1\ .$$

Furthermore, one obtains for $i = 2, 4, \ldots, k-5$ and $l_i < j - 2\,l_{k-2} \leq l_{i+1} + 1$ as well as for $i = 3, 5, \ldots, k-4$ and $l_i + 1 < j - 2\,l_{k-2} \leq l_{i+1}$ that

$$\begin{aligned} p_k^j &= p_{k-2}^j + \bar{p}_{k-2}^{j-2l_{k-2}} \leq \left\lfloor \frac{k-i}{2} \right\rfloor + \left\lceil \frac{i+1}{2} \right\rceil = \left\lceil \frac{k}{2} \right\rceil, \\ s_k^j &= s_{k-2}^j + \bar{s}_{k-2}^{j-2l_{k-2}} \leq k - i + i\ . \end{aligned}$$

For the computational cycles $j = 2\,l_{k-2} + l_{k-4}, \ldots, 3\,l_{k-2} + 1$ follows

$$\begin{aligned} p_k^j &= p_{k-2}^j + \bar{p}_{k-2}^{j-2l_{k-2}} \leq 1 + \left\lceil \frac{k-2}{2} \right\rceil = \left\lceil \frac{k}{2} \right\rceil, \\ s_k^j &= s_{k-2}^j + \bar{s}_{k-2}^{j-2l_{k-2}} \leq 2 + k - 3 = k - 1\ . \end{aligned}$$

Now the parallel reversal schedule S_{k-2} is completed. Hence one obtains for $j = 3\,l_{k-2} + 2, \ldots, 2\,l_{k-2} + l_{k-1} - 2$

$$p_k^j = \bar{p}_{k-2}^{j-2l_{k-2}} \leq \left\lceil \frac{k-1}{2} \right\rceil < \left\lceil \frac{k}{2} \right\rceil, \quad s_k^j = 1 + \bar{s}_{k-2}^{j-2l_{k-2}} \leq k-1\ ,$$

where the checkpoint that stores the initial state is taken into account. The vertex of $\bar{S}_{k-2}$ is reached and the second S_{k-2} starts. Because of the identity $2\,l_{k-2} + l_{k-1} - 2 = 2\,l_{k-1} - 2 = l_k - 1$ one has for $j = l_k$ and $j = l_k + 1$

$$\begin{aligned} p_k^j &= \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-3}{2} + 1 < \left\lceil \frac{k}{2} \right\rceil, \\ s_k^j &= 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq k - 3 + 1 + 1 = k - 1 \end{aligned}$$

and therefore

$$p_k^j \ \leq \ \left\lceil \frac{i+1}{2} \right\rceil \quad \text{and} \quad s_k^j \leq i \quad \text{for} \quad l_i < j \leq \ l_{i+1} + 1 \qquad \text{(B.4)}$$

with $i = k - 1$. Moreover, it follows for $j = l_k + 2$

$$\begin{aligned} p_k^j &= \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-3}{2} + 2 = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq 1 + k - 3 + 2 = k \end{aligned} \tag{B.5}$$

as well as for

$$\begin{aligned} j = l_k + 3: \quad p_k^j &= \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-3}{2} + 2 = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq 1 + k - 4 + 2 < k \\ j = l_k + 4: \quad p_k^j &= \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-3}{2} + 2 = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} = 1 + k - 4 + 3 = k \,. \end{aligned} \tag{B.6}$$

One obtains for $i = 5, \ldots, k-4$ and the computational cycles

$$\begin{aligned} &l_k + l_{i-2} + 1 < j \leq l_k + l_i - 1: \\ &\qquad p_k^j = \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-i}{2} + \left\lceil \frac{i}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ &\qquad s_k^j = 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq 1 + k - i + i - 1 = k \\ &j = l_k + l_i: \\ &\qquad p_k^j = \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-i}{2} + \left\lceil \frac{i}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ &\qquad s_k^j = 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq 1 + k - i - 1 + i - 1 < k \\ &j = l_k + l_i + 1: \\ &\qquad p_k^j = \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq \frac{k-i}{2} + \left\lceil \frac{i+1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ &\qquad s_k^j = 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq 1 + k - i - 1 + i = k \,. \end{aligned} \tag{B.7}$$

For the remainder of $\bar{S}_{k-2}$, i.e. $l_k + l_{k-4} + 1 < j \leq l_k + l_{k-2}$, follows that

$$\begin{aligned} p_k^j &= \bar{p}_{k-2}^{j-l_{k-1}} + p_{k-2}^{j-l_k+1} \leq 1 + \left\lceil \frac{k-2}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= 1 + \bar{s}_{k-2}^{j-l_{k-1}} + s_{k-2}^{j-l_k+1} \leq 1 + 2 + k - 3 = k \,. \end{aligned} \tag{B.8}$$

For the next l_{k-2} computational cycles one processor that performs the forward sweep from the initial state to state l_{k-2} must be taken into account apart from the checkpoint storing the initial state 0. Hence one has for the computational cycles $l_k + l_{k-2} < j \leq 3\, l_{k-1} - 2 = l_k + l_{k-1} - 1$

$$\begin{aligned} p_k^j &= 1 + p_{k-2}^{j-l_k+1} \leq 1 + \left\lceil \frac{k-1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= 2 + s_{k-2}^{j-l_k+1} \leq 2 + k - 2 = k \,. \end{aligned} \tag{B.9}$$

The vertex of the second S_{k-2} is reached, where a third parallel reversal schedule S_{k-2} is placed. Furthermore, one checkpoint stores still the initial state 0.

Therefore one obtains for

$$\begin{aligned}
&j = l_k + l_{k-1} \text{ and } j = l_k + l_{k-1} + 1: \\
&\qquad p_k^j = p_{k-2}^{j-l_k+1} + p_{k-2}^{j-l_k-l_{k-1}+1} \leq \left\lceil \frac{k-2}{2} \right\rceil + 1 = \left\lceil \frac{k+1}{2} \right\rceil, \\
&\qquad s_k^j = 1 + s_{k-2}^{j-l_k+1} + s_{k-2}^{j-l_k-l_{k-1}+1} \leq 1 + k - 2 + 1 = k \\
&j = l_k + l_{k-1} + 2 \text{ and } j = l_k + l_{k-1} + 3: \\
&\qquad p_k^j = p_{k-2}^{j-l_k+1} + p_{k-2}^{j-l_k-l_{k-1}+1} \leq \left\lfloor \frac{k-3}{2} \right\rfloor + 2 = \left\lceil \frac{k+1}{2} \right\rceil, \\
&\qquad s_k^j = 1 + s_{k-2}^{j-l_k+1} + s_{k-2}^{j-l_k-l_{k-1}+1} \leq 1 + k - 3 + 2 = k \,.
\end{aligned} \tag{B.10}$$

Moreover one has for the computational cycles $l_k + l_{k-1} + l_{i-2} < j \leq l_k + l_{k-1} + l_i$ with $i = 5, 7, \ldots, 2\lfloor \frac{k-2}{2} \rfloor - 1$

$$\begin{aligned}
p_k^j &= p_{k-2}^{j-l_k+1} + p_{k-2}^{j-l_k-l_{k-1}+1} \leq \left\lfloor \frac{k-i}{2} \right\rfloor + \left\lceil \frac{i}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\
s_k^j &= 1 + s_{k-2}^{j-l_k+1} + s_{k-2}^{j-l_k-l_{k-1}+1} \leq 1 + k - i + i - 1 = k
\end{aligned} \tag{B.11}$$

and for $l_k + l_{k-1} + l_{2\lfloor \frac{k-2}{2} \rfloor - 1} < j \leq l_k + l_{k-1} + l_{k-2}$

$$\begin{aligned}
p_k^j &= p_{k-2}^{j-l_k+1} + p_{k-2}^{j-l_k-l_{k-1}+1} \leq 1 + \left\lceil \frac{k-2}{2} \right\rceil < \left\lceil \frac{k+1}{2} \right\rceil, \\
s_k^j &= 1 + s_{k-2}^{j-l_k+1} + s_{k-2}^{j-l_k-l_{k-1}+1} \leq 1 + 2 + k - 3 = k \,.
\end{aligned} \tag{B.12}$$

Hence, the second parallel reversal schedule S_{k-2} is completed. From now on one processor is applied to perform a forward sweep from the initial state 0 to state $l_{k-1} - 1$ where the second parallel reversal schedule S_{k-2} starts. Furthermore, one checkpoint stores still the initial state 0. Therefore, one obtains for $l_k + l_{k-1} + l_{k-2} < j \leq 2\, l_k$

$$\begin{aligned}
p_k^j &= 1 + p_{k-2}^{j-l_k-l_{k-1}+1} \leq 1 + \left\lceil \frac{k-1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\
s_k^j &= 2 + s_{k-2}^{j-l_k-l_{k-1}+1} \leq 2 + k - 2 = k \,.
\end{aligned} \tag{B.13}$$

Finally Equations (B.5) up to (B.13) yield with $i = k$

$$p_k^j \leq \left\lceil \frac{i+1}{2} \right\rceil \quad \text{and} \quad s_k^j \leq i \quad \text{for} \quad l_i + 1 < j \leq l_{i+1} \,. \tag{B.14}$$

Subsequently, it follows for

$$\begin{aligned}
j = 2\, l_k + 1: \quad c_k^j &= 1 + c_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lfloor \frac{k-2}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor, \\
p_k^j &= 1 + p_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lceil \frac{k-2}{2} \right\rceil = \left\lceil \frac{k}{2} \right\rceil, \\
s_k^j &= 2 + s_{k-2}^{j-l_k-l_{k-1}+1} = 2 + k - 2 = k
\end{aligned} \tag{B.15}$$

as well as for

$$
\begin{aligned}
j = 2\,l_k + 2 \;\text{ and }\; j = 2\,l_k + 3: \quad & c_k^j = 1 + c_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lfloor \frac{k-2}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor, \\
& p_k^j = 1 + p_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lfloor \frac{k-3}{2} \right\rfloor = \left\lfloor \frac{k-1}{2} \right\rfloor, \\
& s_k^j = 2 + s_{k-2}^{j-l_k-l_{k-1}+1} = k-1 \qquad\qquad (B.16) \\
j = 2\,l_k + 4: \quad & c_k^j = 1 + c_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lfloor \frac{k-2}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor, \\
& p_k^j = 1 + p_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lfloor \frac{k-5}{2} \right\rfloor = \left\lfloor \frac{k-3}{2} \right\rfloor, \\
& s_k^j = 2 + s_{k-2}^{j-l_k-l_{k-1}+1} = k-2\,.
\end{aligned}
$$

Moreover, one has for the remainder of the third parallel reversal schedule S_{k-2}

$$
\begin{aligned}
& c_k^j = 1 + c_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \left\lfloor \frac{k-i+1}{2} \right\rfloor = \left\lfloor \frac{k-i+3}{2} \right\rfloor, \\
& p_k^j \le 1 + p_{k-2}^{j-l_k-l_{k-1}+1} \le 1 + \left\lfloor \frac{k-i}{2} \right\rfloor = \left\lfloor \frac{k-i+2}{2} \right\rfloor, \qquad (B.17) \\
& s_k^j \le 2 + s_{k-2}^{j-l_k-l_{k-1}+1} \le k-i+2
\end{aligned}
$$

with $2\,l_k + l_{i-2} < j - 1 \le 2\,l_k + l_i$, $i = 5, 7, \ldots, 2\lfloor \frac{k-2}{2} \rfloor - 1$, as well as

$$
\begin{aligned}
& c_k^j = 1 + c_{k-2}^{j-l_k-l_{k-1}+1} = 2 = \left\lfloor \frac{k-i+3}{2} \right\rfloor, \\
& p_k^j \le 1 + p_{k-2}^{j-l_k-l_{k-1}+1} = 2 = \left\lfloor \frac{k-i+2}{2} \right\rfloor, \qquad (B.18) \\
& s_k^j \le 2 + s_{k-2}^{j-l_k-l_{k-1}+1} = 4 = k-i+2
\end{aligned}
$$

for $2\,l_k + l_{i-2} < j - 1 \le 2\,l_k + l_i$ and $i = k-2 = 2\lfloor \frac{k}{2} \rfloor - 1$ because $2\lfloor \frac{k-2}{2} \rfloor - 1 = k-4$. Now the third parallel reversal schedule S_{k-2} is also completed. Because of $l_k + 2\,l_{k-1} + l_{k-2} - 1 = 2\,l_k + l_{k-2}$ in the following one has to take into account only the checkpoint storing the initial state 0 and the processor performing the forward sweep to state $l_k - l_{k-2}$ necessary to start there the third S_{k-2}. Hence one obtains

$$
c_k^j = 1, \quad p_k^j = 1, \quad \text{and} \quad s_k^j = 2 \qquad (B.19)
$$

for $2\,l_k + l_{2\lfloor \frac{k}{2} \rfloor - 1} + 1 < j \le 3\,l_k + 1$. Combining Equations (B.2) up to (B.4) and Equations (B.14) up to (B.19) it is proven that the resource profile $R(S_k)$ fulfils Equation (4.27).

It is left to show that Equation (B.1) holds for the resource profile $R(\bar{S}_k)$ constructed by the appropriate shortening of the parallel reversal schedule S_k. Due to the shortening $\bar{S}_{k-2}$ is placed at the vertex of S_k instead of S_{k-2}. Nevertheless, the first $2\,l_k - 2$ computational cycles are taken over from S_k without

any changes. Hence one has

$$\begin{aligned}
&\bar{c}_k^j = c_k^j = 0, \;\; \bar{p}_k^j = p_k^j = 1, \qquad \bar{s}_k^j = s_k^j = 1 \qquad j = 1, 2 \\
&\qquad\qquad \bar{p}_k^j = p_k^j \le \lceil \tfrac{i+1}{2} \rceil, \;\; \bar{s}_k^j = s_k^j \le i \quad \begin{cases} l_i < j \le l_{i+1}{+}1, \; i \text{ even} \\ l_i{+}1 < j \le l_{i+1}, \; i \text{ odd} \\ i = 2, \ldots, k-1 \end{cases} \\
&\qquad\qquad \bar{p}_k^j = p_k^j \le \tfrac{k+1}{2}, \qquad \bar{s}_k^j = s_k^j \le k, \qquad l_k + 1 < j \le 2\,l_k - 2\,.
\end{aligned} \tag{B.20}$$

The second parallel reversal schedule S_{k-2} is completed because of the inequality $7\,l_{k-2} - 1 < 2\,l_k - 2$. There is one processor running from the initial state 0 to state $l_k - 2l_{k-2}$ during the computational cycles $2\,l_k - 1 \le j < 2\,l_k + l_{k-2}$ to start the second S_{k-2} in time. Therefore one has with $2\,l_k = l_{k+1}$ for

$$\begin{aligned}
l_{k+1} - 1 \le j \le l_{k+1} + 1: \;\; \bar{c}_k^j &= 1 + \bar{c}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-3}{2} = \frac{k-1}{2}, \\
\bar{p}_k^j &= 1 + \bar{p}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-3}{2} = \frac{k-1}{2}, \\
\bar{s}_k^j &= 2 + \bar{s}_{k-2}^{j-l_k-l_{k-1}+1} = k - 1 \\
l_{k+1} + 2 \le j \le l_{k+1} + 3: \;\; \bar{c}_k^j &= 1 + \bar{c}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-5}{2} = \frac{k-3}{2}, \\
\bar{p}_k^j &= 1 + \bar{p}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-3}{2} = \frac{k-1}{2}, \\
\bar{s}_k^j &= 2 + \bar{s}_{k-2}^{j-l_k-l_{k-1}+1} = k - 2
\end{aligned} \tag{B.21}$$

as well as for $i = 5, 7, \ldots, k-4$ and

$$\begin{aligned}
&l_{k+1} + l_{i-2} < j \;\le l_{k+1} + l_i - 2: \\
&\qquad \bar{c}_k^j = 1 + \bar{c}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-i}{2} = \frac{k-i+2}{2}, \\
&\qquad \bar{p}_k^j = 1 + \bar{p}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-i}{2} = \frac{k-i+2}{2}, \\
&\qquad \bar{s}_k^j = 2 + \bar{s}_{k-2}^{j-l_k-l_{k-1}+1} = k - i + 2 \\
&l_{k+1} + l_i - 2 < j \le l_{k+1} + l_i: \\
&\qquad \bar{c}_k^j = 1 + \bar{c}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-i-2}{2} = \frac{k-i}{2}, \\
&\qquad \bar{p}_k^j = 1 + \bar{p}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + \frac{k-i}{2} = \frac{k-i+2}{2}, \\
&\qquad \bar{s}_k^j = 2 + \bar{s}_{k-2}^{j-l_k-l_{k-1}+1} = k - i + 1\,.
\end{aligned} \tag{B.22}$$

Furthermore, one obtains for $l_{k+1} + l_{k-4} < j \le l_{k+1} + l_{k-2} - 2$ and $i = k-2$

$$\begin{aligned}
\bar{c}_k^j &= 1 + \bar{c}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + 1 = 2 = \frac{k-i+2}{2}, \\
\bar{p}_k^j &= 1 + \bar{p}_{k-2}^{j-l_k-l_{k-1}+1} = 1 + 1 = 2 = \frac{k-i+2}{2}, \quad \text{and} \\
\bar{s}_k^j &= 2 + \bar{s}_{k-2}^{j-l_k-l_{k-1}+1} = k - i + 2\,.
\end{aligned} \tag{B.23}$$

The second parallel reversal schedule $\bar{S}_{k-2}$ is completed. To make the execution of $\bar{S}_{k-2}$ possible in the next computational cycles one processor has to perform a forward sweep from the initial state 0 to state $l_k - l_{k-2} + 1$. Therefore one finds

$$\begin{aligned} \bar{c}_k^j &= 1 = \frac{k-i}{2}, \quad \bar{p}_k^j = 1 + 1 = 2 = \frac{k-i+2}{2}, \\ \bar{s}_k^j &= 1 + 2 = k - i + 1 \end{aligned} \tag{B.24}$$

for $l_{k+1} + l_i - 2 < j \leq l_{k+1} + l_i$ and $i = k - 2$. The forward sweep from the initial state 0 to state $l_k - 2\,l_{k-2}$ necessary to start the second parallel reversal schedule S_{k-2} is completed. This yields

$$\bar{c}_k^j = 1, \;\; \bar{p}_k^j = 1, \quad \text{and} \quad \bar{s}_k^j = 2 \quad \text{for} \quad l_{k+1} + l_{k-2} < j \leq 3\,l_k - 1\;. \tag{B.25}$$

Equations (B.20) up to (B.25) show that $R(\bar{S}_k)$ fulfils Equation (B.1). Therefore everything is proven for odd k.

Now suppose that k is even. Once more S_k takes the first l_k computational cycles over from S_{k-1} without any changes. Hence one has

$$\begin{aligned} c_k^j = c_{k-1}^j = 0, \;\; p_k^j = p_{k-1}^j = 1, \qquad s_k^j = s_{k-1}^j = 1 \quad & j = 1, 2 \\ p_k^j = p_{k-1}^j \leq \lceil \tfrac{i+1}{2} \rceil, \;\; s_k^j = s_{k-1}^j \leq i \quad & \begin{cases} l_i < j \leq l_{i+1} + 1, i \text{ even} \\ l_i + 1 < j \leq l_{i+1}, i \text{ odd} \\ i = 2, \ldots, k-1\;. \end{cases} \end{aligned} \tag{B.26}$$

Subsequently the second parallel reversal schedule S_{k-1} is placed at the vertex of the first one to enable the reversal without any interruption. Therefore it follows for

$$\begin{aligned} j = l_k + 1, l_k + 2: \;\; p_k^j &= p_{k-1}^j + p_{k-1}^{j-l_k} \leq \left\lceil \frac{k-1}{2} \right\rceil + 1 = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= s_{k-1}^j + s_{k-1}^{j-l_k} \leq k - 1 + 1 = k \\ j = l_k + 3, l_k + 4: \;\; p_k^j &= p_{k-1}^j + p_{k-1}^{j-l_k} \leq \left\lfloor \frac{k-2}{2} \right\rfloor + 2 = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= s_{k-1}^j + s_{k-1}^{j-l_k} \leq k - 2 + 2 = k\;. \end{aligned} \tag{B.27}$$

Moreover, one has for $l_k + l_{i-2} + 1 < j \leq l_k + l_i + 1$ with $i = 5, 7, \ldots, 2\lfloor \frac{k-1}{2} \rfloor - 1$

$$\begin{aligned} p_k^j &= p_{k-1}^j + p_{k-1}^{j-l_k} \leq \left\lfloor \frac{k-i+1}{2} \right\rfloor + \left\lceil \frac{i}{2} \right\rceil \leq \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= s_{k-1}^j + s_{k-1}^{j-l_k} \leq k - i + 1 + i - 1 = k \end{aligned} \tag{B.28}$$

and for $l_k + l_{2\lfloor \frac{k-1}{2} \rfloor - 1} < j - 1 \leq l_k + l_{k-1}$

$$\begin{aligned} p_k^j &= p_{k-1}^j + p_{k-1}^{j-l_k} \leq 1 + \left\lceil \frac{k-1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ s_k^j &= s_{k-1}^j + s_{k-1}^{j-l_k} \leq 2 + k - 2 = k\;. \end{aligned} \tag{B.29}$$

The first parallel reversal schedules S_{k-1} is completed. Hence from now on only the checkpoint storing the initial state 0 must be taken into account and one obtains for $l_k + l_{k-1} + 1 < j \leq 2\,l_k$

$$p_k^j = p_{k-1}^{j-l_k} \leq \left\lceil \frac{k}{2} \right\rceil < \left\lceil \frac{k+1}{2} \right\rceil, \qquad s_k^j = 1 + s_{k-1}^{j-l_k} \leq 1 + k - 1 = k\;. \tag{B.30}$$

Therefore it is shown by Equations (B.27) up to (B.30) that

$$p_k^j \le \left\lceil \frac{k+1}{2} \right\rceil \quad \text{and} \quad s_k^j \le k \quad \text{for} \quad l_k < j \le l_{k+1} + 1 \,. \tag{B.31}$$

Furthermore, it follows for

$$\begin{aligned}
&j = 2\,l_k + 1: \; c_k^j = 1 + c_{k-1}^{j-l_k} = 1 + \left\lfloor \frac{k-1}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor, \\
&\quad p_k^j = p_{k-1}^{j-l_k} = \left\lceil \frac{k-1}{2} \right\rceil = \left\lceil \frac{k}{2} \right\rceil, \quad s_k^j = 1 + s_{k-1}^{j-l_k} = 1 + k - 1 = k \\
&j = 2\,l_k + 2 \text{ and } j = 2\,l_k + 3: \; c_k^j = 1 + c_{k-1}^{j-l_k} = 1 + \left\lfloor \frac{k-1}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor, \\
&\quad p_k^j = p_{k-1}^{j-l_k} = \left\lfloor \frac{k-2}{2} \right\rfloor = \left\lfloor \frac{k-1}{2} \right\rfloor, \quad s_k^j = 1 + s_{k-1}^{j-l_k} = k - 1 \\
&j = 2\,l_k + 4: \; c_k^j = 1 + c_{k-1}^{j-l_k} = 1 + \left\lfloor \frac{k-1}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor, \\
&\quad p_k^j = p_{k-1}^{j-l_k} = \left\lfloor \frac{k-4}{2} \right\rfloor = \left\lfloor \frac{k-3}{2} \right\rfloor, \quad s_k^j = 1 + s_{k-1}^{j-l_k} = k - 2
\end{aligned} \tag{B.32}$$

as well as for $2\,l_k + l_{i-2} < j - 1 \le 2\,l_k + l_i$ with $i = 5, 7, \ldots, 2\lfloor \frac{k-1}{2} \rfloor - 1$

$$\begin{aligned}
&c_k^j = 1 + c_{k-1}^{j-l_k} = 1 + \left\lfloor \frac{k-i+2}{2} \right\rfloor = \left\lfloor \frac{k-i+3}{2} \right\rfloor, \\
&p_k^j = p_{k-1}^{j-l_k} \le \left\lfloor \frac{k-i+1}{2} \right\rfloor = \left\lfloor \frac{k-i+2}{2} \right\rfloor, \quad s_k^j = 1 + s_{k-1}^{j-l_k} \le k - i + 2 \,.
\end{aligned} \tag{B.33}$$

For the remainder of the second parallel reversal schedule S_{k-1} one has for $2\,l_k + l_{i-2} < j - 1 \le 2\,l_k + l_i$ and $i = k - 1 = 2\lfloor \frac{k-1}{2} \rfloor - 1$

$$\begin{aligned}
&c_k^j = 1 + c_{k-1}^{j-l_k} = 2 = \left\lfloor \frac{k-i+3}{2} \right\rfloor, \\
&p_k^j = p_{k-1}^{j-l_k} = 1 = \left\lfloor \frac{k-i+2}{2} \right\rfloor, \quad s_k^j = 1 + s_{k-1}^{j-l_k} = 3 = k - i + 2
\end{aligned} \tag{B.34}$$

because $2\lfloor \frac{k-1}{2} \rfloor - 1 = k - 3$. The second parallel reversal schedule S_{k-1} is completed and in the remaining computational cycles one processor has to perform a forward sweep from the initial state 0 to state l_{k-1} in order to start the second S_{k-1}. Therefore one has

$$c_k^j = 1, \quad p_k^j = 1, \quad \text{and} \quad s_k^j = 2 \tag{B.35}$$

for $2\,l_k + l_{k-1} + 1 < j \le 2\,l_k + l_k + 1$. Combining Equations (B.26), (B.31), and (B.32) up to (B.35) it is shown that the resource profile $R(S_k)$ fulfils Equation (4.27) for even k. Summarizing the above the assertion is proven for all $k \in \mathbb{N}$. ∎

In order to construct feasible parallel reversal schedules S_k for the case $\bar{t} = 2$ and $\hat{t} \in \mathbb{N}$, $\hat{t} > 2$, a similar approach as in the proof of Theorem 4.6 is applied:

Proof of Theorem 4.9, Page 94:
First, assume $1 \leq k < 2 + \hat{t}/2$. Appropriate parallel reversal schedules S_k can be constructed according to the ones described in the beginning of the proof of Theorem 4.6. For each physical step i, $0 \leq i \leq l_k - 1 = k - 1$ one processor performs a forward sweep from the initial state 0 to state i, one recording step $\hat{F}_i$, and one reverse step $\bar{F}_i$. Then it is possible to control all k processors in order to calculate the reversal without any interruption (see Fig. B.5). It is

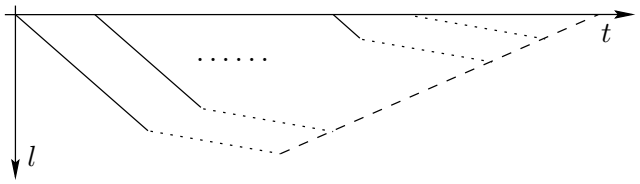


Figure B.5: Reversal Schedule S_k, where $k < 2 + \hat{t}/2$

easy to check that one obtains the resource profile

$$
\begin{array}{llll}
c_k^j = 0, & p_k^j = i, & s_k^j = i & j = 2i-1, 2i,\ 1 \leq i \leq k-1 \\
c_k^j = 0, & p_k^j = k, & s_k^j = k & j = 2k-1 \\
c_k^j = 0, & p_k^j = k, & s_k^j = k & 2k-1 < j \leq 2+\hat{t} \\
c_k^j = 1, & p_k^j = k-i, & s_j^k = k-i+1 & \begin{cases} 2+\hat{t}+3(i-1) < j \leq 2+\hat{t}+3i \\ 1 \leq i \leq k-1\ . \end{cases}
\end{array}
$$

Hence the assertion is proven for $1 \leq k < 2 + \hat{t}/2$.

Second, for $k \geq 2 + \hat{t}/2$ parallel reversal schedules S_k needing no more than k resources are constructed for the reversal of l_k physical steps. Furthermore, the number of processors required by S_k is limited above by p_k^b. For that let $l_{k,1}$ and $l_{k,2}$ denote the number of physical steps that can be reversed with up to k processors and checkpoints applying the parallel reversal schedules developed in Theorem 4.7 and Theorem 4.8, respectively. Moreover the corresponding feasible reversal schedules constructed in these theorems are denoted by $S_{k,1}$ and $S_{k,2}$. Based on Equation (4.28) appropriate parallel reversal schedules S_k will be constructed using the parallel reversal schedules $S_{k-(\hat{t}-1)/2,1}$ and $S_{k-(\hat{t}-2)/2,2}$, respectively.

First, assume that $\hat{t}$ is odd. Consider for an arbitrary $k > 2 + \hat{t}/2$ the feasible parallel reversal schedule $S_{\tilde{k},1}$ with $\tilde{k} \equiv k - (\hat{t}-1)/2$. Take the first $2\,l_{\tilde{k},1} + 1$ computational cycles of $S_{\tilde{k},1}$ as constructed in Theorem 4.7 and transform them into the first $2\,l_{\tilde{k},1} + \hat{t}$ computational cycles of a parallel reversal schedule $\bar{S}_k$ such that the given value of $\hat{t}$ is taken into account. This modification corresponds to shifting the reverse sweep to the right as depicted already in Fig. 4.25 for the case $k = 7$, $\hat{t} = 3$ and $\bar{t} = 1$. Everything else, especially the checkpoint writings, remain unchanged. Hence, it follows that during each computational cycle j with $1 \leq j \leq 2\,l_{\tilde{k},1}$ of $\bar{S}_k$ there are no more than $(\hat{t}+1)/2$ processors performing recording steps instead of at most one processor in the corresponding computational cycle of $S_{\tilde{k},1}$. Moreover for each $m = 1, \ldots, \hat{t} - 1$

the computational cycle $j = 2\,l_{\tilde{k},1} - \hat{t} + 1 + m$ of $S_{\tilde{k},1}$ applies one processor for a reverse step if j is even and two processors for a recording step and a reverse step if j is odd. In the corresponding computational cycle $j = 2\,l_{\tilde{k},1} + m$ of $S_{\tilde{k},1}$ no more than $(\hat{t}+1)/2 - 1 - \lfloor (m-1)/2 \rfloor$ processors are needed only for the recording steps if j is even and no more than $(\hat{t}+1)/2 - \lfloor (m-1)/2 \rfloor$ processors if j is odd. No reverse step has started yet. Therefore, the number of processors needed and the total number of applied processors and checkpoints increase during each computational cycle $j = 2\,l_{\tilde{k},1} + m$ with $m = 1, \ldots, \hat{t} - 1$ of $\bar{S}_k$ by $(\hat{t}-3)/2 - \lfloor (m-1)/2 \rfloor$ compared to the corresponding computational cycle of $S_{\tilde{k},1}$. Moreover, the computational cycle $2\,l_{\tilde{k},1}+1$ of the parallel reversal schedule $S_{\tilde{k},1}$ constructed in Theorem 4.7 corresponds exactly to the computational cycle $2\,l_{\tilde{k},1} + \hat{t}$ of $\bar{S}_k$. This yields the following resource profile for the first $2\,l_{\tilde{k},1} + \hat{t}$ computational cycles of the parallel reversal schedule $\bar{S}_k$

$$
\begin{array}{lll}
c_k^j = 0, \quad p_k^j = i, \quad\quad s_k^j = i & & j = 2i-1, 2i,\ 1 \le i \le (\hat{t}+1)/2 \\
c_k^j = 0, \quad p_k^j = \frac{\hat{t}+1}{2} + 1,\ s_k^j = \frac{\hat{t}+1}{2} + 1 & & j = \hat{t}+2 \\
\left.\begin{array}{l} p_k^j \le p_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-1}{2}, \\ s_k^j \le s_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-1}{2} \end{array}\right\} & & \hat{t}+2 < j \le 2\,l_{\tilde{k},1} \\
\left.\begin{array}{l} p_k^j \le p_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-3}{2} - \lfloor \frac{m-1}{2} \rfloor, \\ s_k^j \le s_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-3}{2} - \lfloor \frac{m-1}{2} \rfloor \end{array}\right\} & & j = 2\,l_{\tilde{k},1} + m,\ 1 \le m \le \hat{t}-1 \\
p_k^j = p_{\tilde{k},1}^{j-\hat{t}+1}, \quad s_k^j = s_{\tilde{k},1}^{j-\hat{t}+1} & & j = 2\,l_{\tilde{k},1} + \hat{t}\,.
\end{array}
\tag{B.36}
$$

Therefore one has for the computational cycles $j = 1, \ldots, 2\,l_{\tilde{k},1}$

$$
\begin{aligned}
p_k^j &\le \max\left\{ \frac{\hat{t}+1}{2} + 1, \max_{\hat{t}+1 < j \le 2\,l_{\tilde{k},1}} \left\{ p_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-1}{2} \right\} \right\} \\
&\le \max\left\{ \frac{\hat{t}+1}{2} + 1, \left\lceil \frac{\tilde{k}+1}{2} \right\rceil + \frac{\hat{t}-1}{2} \right\} \le \left\lceil \frac{k+1}{2} \right\rceil + \left\lceil \frac{\hat{t}-1}{4} \right\rceil
\end{aligned}
\tag{B.37}
$$

as well as

$$
\begin{aligned}
s_k^j &\le \max\left\{ \frac{\hat{t}+1}{2} + 1, \max_{\hat{t}+1 < j \le 2\,l_{\tilde{k},1}} \left\{ s_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-1}{2} \right\} \right\} \\
&\le \max\left\{ \frac{\hat{t}+1}{2} + 1, k - \frac{\hat{t}-1}{2} + \frac{\hat{t}-1}{2} \right\} = k\,.
\end{aligned}
\tag{B.38}
$$

Moreover for each computational cycle $j = 2\,l_{\tilde{k},1} + m$ with $1 \le m \le \hat{t} - 1$ follows that

$$
\begin{aligned}
p_k^j &= p_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-3}{2} - \left\lfloor \frac{m-1}{2} \right\rfloor \le \left\lceil \frac{k+1}{2} \right\rceil + \left\lceil \frac{\hat{t}-1}{4} \right\rceil - 1 - \left\lfloor \frac{m-1}{2} \right\rfloor, \\
s_k^j &= s_{\tilde{k},1}^{j-\hat{t}+1} + \frac{\hat{t}-3}{2} - \left\lfloor \frac{m-1}{2} \right\rfloor \le k - 1 - \left\lfloor \frac{m-1}{2} \right\rfloor\,.
\end{aligned}
\tag{B.39}
$$

It is possible to put the first $2\,l_{\tilde{k},1} + \hat{t}$ computational cycles of $\bar{S}_k$ and the computational cycles j with $j = 2\,l_{\tilde{k},1} + 2, \ldots, 3\,l_{\tilde{k},1}$ of $S_{\tilde{k},1}$ together to form a complete parallel reversal schedule $\bar{S}_k$ for $l_{\tilde{k},1}$ physical steps and given $\hat{t}$. This can be done because the computational cycle $2\,l_{\tilde{k},1} + \hat{t}$ of $\bar{S}_k$ is equal to the computational cycle $2\,l_{\tilde{k},1} + 1$ of $S_{\tilde{k},1}$. The resulting parallel reversal schedule corresponds to the one shown in Fig. 4.26 for $k = 7$, $\hat{t} = 3$, and $\bar{t} = 1$. It follows for the computational cycles j with $j = 2\,l_{\tilde{k},1} + \hat{t} + 1, \ldots, 3\,l_{\tilde{k},1} + \hat{t} - 1$ of $\bar{S}_k$ that

$$p_k^j = p_{\tilde{k},1}^{j-\hat{t}+1} \leq \left\lceil \frac{k-(\hat{t}-1)/2+1}{2} \right\rceil \text{ and } s_k^j = s_{\tilde{k},1}^{j-\hat{t}+1} \leq k - (\hat{t}-1)/2\,. \quad \text{(B.40)}$$

From the Equations (B.39) and (B.40) it is possible to conclude that for each $m = 1, 3, \ldots, \hat{t} - 2$ one processor is available for all computational cycles j with $j = 2\,l_{\tilde{k},1} + m, \ldots, 3\,l_{\tilde{k},1} + \hat{t} - 1$. This processor can be used in the following way to increase the number of physical steps that are reversed and hence to create the desired parallel reversal schedule S_k. The free processor performs the reverse step $\bar{F}_{l_{\tilde{k},1}+(m-1)/2}$ during the computational cycles $2\,l_{\tilde{k},1} + m$ and $2\,l_{\tilde{k},1} + m + 1$. Then the recording step $\hat{F}_{l_{\tilde{k},1}+(m-1)/2}$ is evaluated by the free processor in the computational cycles $2\,l_{\tilde{k},1} + m + 2, \ldots, 2\,l_{\tilde{k},1} + m + \hat{t} + 1$. In the computational cycles $2\,l_{\tilde{k},1} + m + \hat{t} + 2, \ldots, 3\,l_{\tilde{k},1} + m + \hat{t} + 2 + (m+1)/2$ the forward sweep from the initial state 0 to state $l_{\tilde{k},1} + (m-1)/2$ is performed by the free processor. This extension is possible $(\hat{t}-1)/2$ times. Obviously, the checkpoint writing copying the initial state 0 has to be done in the computational cycle $3\,l_{\tilde{k},1} + \hat{t} - 1 + 3(\hat{t}-1)/2$ instead of $3\,l_{\tilde{k},1}$. The total number of physical steps that can be reversed with S_k equals $l_{\tilde{k},1} + (\hat{t}-1)/2 = l_k$ because of Equation (4.28). The number of computational cycles in S_k is given by $3\,l_{\tilde{k},1} + \hat{t} - 1 + 3(\hat{t}-1)/2 = 3\,l_k + \hat{t} - 1$. Furthermore, it follows from the Equations (B.37) up to (B.40), and

$$\left\lceil \frac{k-(\hat{t}-1)/2+1}{2} \right\rceil + \frac{\hat{t}-1}{2} = \left\lceil \frac{k+1}{2} \right\rceil + \left\lceil \frac{\hat{t}-1}{4} \right\rceil$$

that the number of processors needed is not greater than p_k^b. Moreover it is shown that the number of applied checkpoints and processors at any time does not exceed k. Therefore the assertion is proven if $\hat{t}$ is odd.

Now suppose that $\hat{t}$ is even and consider for an arbitrary $k \geq 2 + \hat{t}/2$ the parallel reversal schedule $S_{\tilde{k},2}$ with $\tilde{k} \equiv k - (\hat{t}-2)/2$. Change the first $2\,l_{\tilde{k},2} + 2$ computational cycles of $S_{\tilde{k},2}$ as constructed in the proof of Theorem 4.8 in order to take the given value of $\hat{t}$ into account. This corresponds to shifting the reverse sweep to the right as depicted already in Fig. 4.25 for $k = 7$, $\hat{t} = 3$ and $\bar{t} = 1$. Everything else, especially the checkpoint writings, remain unchanged. Now these $2\,l_{\tilde{k},2} + \hat{t}$ computational cycles form the first part of a parallel reversal schedule denoted by $\bar{S}_k$. During each computational cycle j with $1 \leq j \leq 2\,l_{\tilde{k},2}$ of $\bar{S}_k$ there are no more than $\hat{t}/2$ processors that perform recording steps instead of at most one processor that performs a recording step in the corresponding computational cycle of $S_{\tilde{k},2}$. Furthermore, for each $m = 1, \ldots, \hat{t} - 1$ the computational cycle $j = 2\,l_{\tilde{k},2} - \hat{t} + 1 + m$ of $S_{\tilde{k},2}$ applies two processors for a

recording step and a reverse step. In the corresponding computational cycle $j = 2\,l_{\tilde{k},2} + m$ of $\bar{S}_k$ no more than $\hat{t}/2 + 1 - \lceil m/2 \rceil$ processors are needed only for the recording. No reverse step has started yet. Therefore, the number of processors needed and the total number of applied processors and checkpoints increase by $\hat{t}/2 - 1 - \lceil m/2 \rceil$ during each computational cycle $j = 2\,l_{\tilde{k},2} + m$ with $m = 1, \ldots, \hat{t} - 1$ of $\bar{S}_k$ compared to the corresponding computational cycle of $S_{\tilde{k},2}$. Moreover, the computational cycles $2\,l_{\tilde{k},2} + 1$ and $2\,l_{\tilde{k},2} + 2$ of the parallel reversal schedule $S_{\tilde{k},2}$ constructed in Theorem 4.8 are identical to the computational cycles $2\,l_{\tilde{k},2} + \hat{t} - 1$ and $2\,l_{\tilde{k},2} + \hat{t}$ of $\bar{S}_k$. Therefore the resource profile $R(\bar{S}_k)$ of the parallel reversal schedule $\bar{S}_k$ fulfils

$$\begin{array}{lll}
c_k^j = 0, \quad p_k^j = i, \quad s_k^j = i & & j = 2i-1, 2i,\ 1 \le i \le \hat{t}/2+1 \\
\left.\begin{array}{l} p_k^j = p_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2}, \\ s_k^j = s_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} \end{array}\right\} & & \hat{t}+2 < j \le 2\,l_{\tilde{k},2} \\
\left.\begin{array}{l} p_k^j = p_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} - \lceil \frac{m}{2} \rceil, \\ s_k^j = s_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} - \lceil \frac{m}{2} \rceil \end{array}\right\} & & j = 2\,l_{\tilde{k},2} + m,\ 1 \le m \le \hat{t}-2 \\
p_k^j = p_{\tilde{k},2}^{j-\hat{t}+1}, \quad s_k^j = s_{\tilde{k},2}^{j-\hat{t}+1} & & j = 2\,l_{\tilde{k},2} + \hat{t} - 1,\ 2\,l_{\tilde{k},2} + \hat{t}\ .
\end{array} \tag{B.41}$$

Hence one obtains for the computational cycles $j = 1, \ldots, 2\,l_{\tilde{k},2}$

$$\begin{aligned}
p_k^j &\le \max\left\{\frac{\hat{t}}{2} + 1, \max_{\hat{t}+1 < j \le 2\,l_{\tilde{k},2}} \left\{ p_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} \right\}\right\} \\
&\le \max\left\{\frac{\hat{t}}{2} + 1, \left\lceil \frac{\tilde{k}+1}{2} \right\rceil + \frac{\hat{t}-2}{2}\right\} \le \left\lceil \frac{k+1}{2} \right\rceil + \left\lceil \frac{\hat{t}-1}{4} \right\rceil, \\
s_k^j &\le \max\left\{\frac{\hat{t}}{2} + 1, \max_{\hat{t}+1 < j \le 2\,l_{\tilde{k},2}} \left\{ s_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} \right\}\right\} \\
&\le \max\left\{\frac{\hat{t}}{2} + 1, k - \frac{\hat{t}-2}{2} + \frac{\hat{t}-2}{2}\right\} = k\ .
\end{aligned} \tag{B.42}$$

Moreover, for each $j = 2\,l_{\tilde{k},2} + m$ with $1 \le m \le \hat{t} - 2$ follows that

$$\begin{aligned}
p_k^j &= p_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} - \left\lceil \frac{m}{2} \right\rceil \le \left\lceil \frac{k+1}{2} \right\rceil + \left\lceil \frac{\hat{t}-1}{4} \right\rceil - \left\lceil \frac{m}{2} \right\rceil, \\
s_k^j &= s_{\tilde{k},2}^{j-\hat{t}+1} + \frac{\hat{t}-2}{2} - \left\lceil \frac{m}{2} \right\rceil \le k - \left\lceil \frac{m}{2} \right\rceil\ .
\end{aligned} \tag{B.43}$$

Because the computational cycle $2\,l_{\tilde{k},2} + \hat{t}$ of $\bar{S}_k$ is equal to the computational cycle $2\,l_{\tilde{k},2} + 2$ of $S_{\tilde{k},2}$ one can put the $2\,l_{\tilde{k},2} + \hat{t}$ computational cycles of $\bar{S}_k$ and the computational cycles j with $j = 2\,l_{\tilde{k},2} + 3, \ldots, 3\,l_{\tilde{k},2} + 1$ of $S_{\tilde{k},2}$ together to form a complete parallel reversal schedule $\bar{S}_k$ for $l_{\tilde{k},2}$ physical steps and given $\hat{t}$. The resulting parallel reversal schedule corresponds to the one shown in Fig. 4.26 for $k = 7$, $\hat{t} = 3$, and $\bar{t} = 1$. It follows for the computational cycles j

with $j = 2\,l_{\tilde{k},2} + \hat{t} + 1, \ldots, 3\,l_{\tilde{k},2} + \hat{t} - 1$ of $\bar{S}_k$ that

$$p_k^j = p_{\tilde{k},2}^{j-\hat{t}+1} \leq \left\lceil \frac{k-(\hat{t}-2)/2+1}{2} \right\rceil \text{ and } s_k^j = s_{\tilde{k},2}^{j-\hat{t}+1} \leq k - (\hat{t}-2)/2\,. \quad \text{(B.44)}$$

Equations (B.43) and (B.44) yield that for each $m = 1, 3, \ldots, \hat{t}-3$ one processor is available in the computational cycles $2\,l_{\tilde{k},2}+m, \ldots, 3\,l_{\tilde{k},2}+\hat{t}-1$. This processor can be used in the following way to increase the number of physical steps that are reversed and hence to create the desired parallel reversal schedule S_k. First, it performs the reverse step $\bar{F}_{l_{\tilde{k},2}+(m-1)/2}$ in the computational cycles $2\,l_{\tilde{k},2}+m$ and $2\,l_{\tilde{k},2}+m+1$. Then the recording step $\hat{F}_{l_{\tilde{k},2}+(m-1)/2}$ is evaluated by the free processor in the computational cycles $2\,l_{\tilde{k},2}+m+2, \ldots, 2\,l_{\tilde{k},2}+m+\hat{t}+1$. The forward sweep from the initial state 0 to state $l_{\tilde{k},2}+(m-1)/2$ is performed using also the available processor during the computational cycles j with $j = 2\,l_{\tilde{k},2}+m+\hat{t}+2, \ldots, 3\,l_{\tilde{k},2}+m+\hat{t}+2+(m+1)/2$. This extension is possible $(\hat{t}-2)/2$ times. Naturally, the checkpoint writing copying the initial state 0 has to be performed in the computational cycle $3\,l_{\tilde{k},2}+\hat{t}-1+3(\hat{t}-2)/2$ instead of $3\,l_{\tilde{k},2}+1$. The total number of physical steps that can be reversed with S_k is given by $l_{\tilde{k},2}+(\hat{t}-2)/2 = l_k$ because of Equation (4.28). The number of computational cycles in S_k equals properly $3\,l_{\tilde{k},2}+\hat{t}-1+3(\hat{t}-2)/2 = 3\,l_k+\hat{t}-1$. Furthermore, it follows from Equations (B.42) up to (B.44), and

$$\left\lceil \frac{k-(\hat{t}-2)/2+1}{2} \right\rceil + \frac{\hat{t}-2}{2} \leq \left\lceil \frac{k+1}{2} \right\rceil + \left\lceil \frac{\hat{t}-1}{4} \right\rceil$$

that the number of processors needed is not greater than p_k^b. Furthermore, it has been shown that the number of resources that are used does not exceed k at any time. Therefore the theorem is proven also if $\hat{t}$ is even, which completes the proof. ∎

B.2 Construction of Parallel Reversal Schedules for $\bar{t} > 2$

First, some auxiliary sequences that are needed in the remaining proofs are considered. Several properties of the sequences are shown. Then two theorems deal with auxiliary parallel reversal schedules $\tilde{S}_k$ for the cases $\hat{t} < \bar{t}$ and $\hat{t} = \bar{t}$. These $\tilde{S}_k$ are applied in in the proofs of two further theorems to construct the desired feasible reversal schedules S_k to reverse l_k physical steps for the cases $\hat{t} \leq \bar{t}$ and $\hat{t} > \bar{t}$.

Lemma B.1 (Some Auxiliary Sequences and Their Properties). *For given values $\bar{t}, \hat{t} \in \mathbb{N}$ with $\bar{t} > 2$ and $\hat{t} \leq \bar{t}$ define for $k \in \mathbb{N}$ the sequences*

$$l_k \equiv \begin{cases} k & \text{if } 0 \leq k \leq 2 \\ l_{k-1} + \bar{t}\,l_{k-2} - \hat{t} + 1 & \text{else} \end{cases} \quad \text{and}$$

$$\tilde{l}_k \equiv \begin{cases} 0 & \text{if } k = 0 \\ l_{k+1} - l_k & \text{if } 1 \leq k \leq 3 \\ \tilde{l}_{k-1} + \bar{t}(l_{k-1} - l_{k-2}) & \text{else} \end{cases}.$$

Consider a fixed $k \in \mathbb{N}$. If $\hat{t} < \bar{t}$ let $\check{l}_j$ be given by

$$\check{l}_j \equiv \begin{cases} 0 & j = 0 \\ \bar{t} & j = 1 \text{ and } k \text{ even} \\ \hat{t}+1 & j = 1 \text{ and } k \text{ odd} \\ \check{l}_{j-1} + \tilde{l}_j & j > 1 \text{ and } k+j \text{ even} \\ \tilde{l}_{j+1} + \hat{t} - 1 & j > 1 \text{ and } k+j \text{ odd} \end{cases} .$$

If $\hat{t} = \bar{t}$ set

$$\check{l}_j \equiv \begin{cases} 0 & j = 0 \\ \bar{t}-2 & j = 1 \text{ and } k \text{ even} \\ \bar{t}+1 & j = 2 \text{ and } k \text{ even} \\ \bar{t}-1 & j = 1 \text{ and } k \text{ odd} \\ 2\bar{t}-1 & j = 2 \text{ and } k \text{ odd} \\ 2\bar{t} & j = 3 \text{ and } k \text{ odd} \\ \check{l}_{j-1} + \tilde{l}_j & j > 3 \text{ and } k+j \text{ even} \\ \tilde{l}_{j+1} + \bar{t} - 1 & j > 2 \text{ and } k+j \text{ odd} \end{cases} .$$

Then for $k \geq 4$ the equalities

$$\tilde{l}_k = l_k - l_{k-1} + \bar{t}(l_{k-1} - l_{k-2}) \quad \text{and} \quad \tilde{l}_{k-1} + \tilde{l}_{k-2} + l_{k-2} = l_k$$

as well as the inequalities

$$2l_{k-1} \leq l_k, \qquad 2\tilde{l}_k \leq \bar{t}\, l_k, \qquad \text{and} \qquad \tilde{l}_k \leq \bar{t}(l_k - l_{k-1})$$

are valid. If $k \in \mathbb{N}$ is fixed one obtains $\check{l}_j \leq \bar{t}\, l_j$ for $1 \leq j \leq k$.

Proof. The assertions will be proven by induction on k and j, respectively. In order to show the equality $\tilde{l}_k = l_k - l_{k-1} + \bar{t}(l_{k-1} - l_{k-2})$ one has for $k = 4$

$$\tilde{l}_4 = \tilde{l}_3 + \bar{t}(l_3 - l_2) = l_4 - l_3 + \bar{t}(l_3 - l_2) .$$

Now assume that $\tilde{l}_{k-1} = l_{k-1} - l_{k-2} + \bar{t}(l_{k-2} - l_{k-3})$ is valid for $k > 4$. Then using the definition of l_k in case of $k > 3$ for $\tilde{l}_k$ follows

$$\begin{aligned} \tilde{l}_k &= \tilde{l}_{k-1} + \bar{t}(l_{k-1} - l_{k-2}) \\ &= l_{k-1} - l_{k-2} + \bar{t}(l_{k-2} - l_{k-3}) + \bar{t}(l_{k-1} - l_{k-2}) \\ &= l_k - l_{k-1} + \bar{t}(l_{k-1} - l_{k-2}) . \end{aligned}$$

The first equality is proven. In order to show the second equality for $k = 4$ and $k = 5$ one has

$$\begin{aligned} \tilde{l}_3 + \tilde{l}_2 + l_2 &= l_4 - l_3 + l_3 - l_2 + l_2 = l_4 \\ \tilde{l}_4 + \tilde{l}_3 + l_3 &= l_4 - l_3 + \bar{t}(l_3 - l_2) + l_4 - l_3 + l_3 = l_5 . \end{aligned}$$

If $\tilde{l}_{j-2} + \tilde{l}_{j-3} + l_{j-3} = l_{j-1}$ holds for $4 \leq j < k$ with $k \geq 6$ one finds that

$$\begin{aligned} \tilde{l}_{k-1} + \tilde{l}_{k-2} + l_{k-2} &= \tilde{l}_{k-2} + \bar{t}(l_{k-2} - l_{k-3}) + \tilde{l}_{k-3} + \bar{t}(l_{k-3} - l_{k-4}) + l_{k-2} \\ &= l_{k-1} - l_{k-3} + \bar{t} l_{k-2} + l_{k-2} - \bar{t} l_{k-4} = l_k \end{aligned}$$

and the second equality is proven. In order to show $2\,l_{k-1} \leq l_k$ one obtains for $k = 4$ and $k = 5$

$$\begin{aligned} 2\,l_3 &= 2\,\bar{t} - 2\,\hat{t} + 6 < 3\,\bar{t} - 2\,\hat{t} + 4 = l_4 \\ 2\,l_4 &= 6\,\bar{t} - 4\,\hat{t} + 8 \leq 6\,\bar{t} - 3\,\hat{t} + 5 + \bar{t}(\bar{t} - \hat{t}) = l_5\ . \end{aligned}$$

Suppose $2\,l_{k-3} \leq l_{k-2}$ is valid for $k \geq 6$. Since $k - 1 > 3$ one has

$$\begin{aligned} & l_k = l_{k-1} + \bar{t}\,l_{k-2} - \hat{t} + 1 \geq 2\,l_{k-1} \quad \Longleftrightarrow \\ & \bar{t}\,l_{k-2} - \hat{t} + 1 \geq l_{k-1} = l_{k-2} + \bar{t}\,l_{k-2} - \hat{t} + 1 \quad \Longleftrightarrow \quad (\bar{t} - 1)l_{k-2} \geq \bar{t}\,l_{k-3}\ . \end{aligned}$$

Because $(\bar{t} - 1)l_{k-2} \geq 2(\bar{t} - 1)l_{k-3} \geq \bar{t}\,l_{k-3}$ the last inequality is true and therefore the assertion proven. Now it will be shown that $2\,\tilde{l}_k \leq \bar{t}\,l_k$ holds. For $k = 4$ one has

$$\begin{aligned} 2\,\tilde{l}_4 &= 2\,\tilde{l}_3 + 2\,\bar{t}(l_3 - l_2) = 2\,\bar{t}^{\,2} + 6\,\bar{t} - 2\,\bar{t}\,\hat{t} - 2\,\hat{t} + 2 \\ &< 3\,\bar{t}^{\,2} + 4\,\bar{t} - 2\,\bar{t}\,\hat{t} = \bar{t}(4 + 3\,\bar{t} - 2\,\hat{t}) = \bar{t}\,l_4 \end{aligned}$$

because $\bar{t} > 2$. For $k > 4$ again the induction principle is used. Assuming that $2\,\tilde{l}_{k-1} \leq \bar{t}\,l_{k-1}$ holds for $k > 4$ one obtains

$$2\,\tilde{l}_k = 2(l_k - l_{k-1}) + 2\,\bar{t}(l_{k-1} - l_{k-2}) = 2\,\bar{t}\,l_{k-1} - 2\,\hat{t} + 2 \leq 2\,\bar{t}\,l_{k-1} \leq \bar{t}\,l_k$$

and the asserted inequality is proven. In order to show $\tilde{l}_k \leq \bar{t}(l_k - l_{k-1})$ first consider for $k = 4$

$$\tilde{l}_4 = 2\,\bar{t} - \hat{t} + 1 + \bar{t}(\bar{t} - \hat{t} + 1) \leq \bar{t}^{\,2} + \bar{t}(\bar{t} - \hat{t} + 1) = \bar{t}(l_4 - l_3)\ .$$

Now assume that $\tilde{l}_{k-1} \leq \bar{t}(l_{k-1} - l_{k-2})$ is true for $k > 4$. This yields

$$\tilde{l}_k = \tilde{l}_{k-1} + \bar{t}(l_{k-1} - l_{k-2}) \leq 2\,\bar{t}(l_{k-1} - l_{k-2}) \leq 2\,\bar{t}\,l_{k-1} \leq \bar{t}\,l_k$$

and the asserted inequality is proven. Finally for fixed k it will be shown that $\check{l}_j \leq \bar{t}\,l_j$ holds for $1 \leq j \leq k$. First assume that $\hat{t} < \bar{t}$. Then one has

$$\check{l}_1 \leq \bar{t} = \bar{t}\,l_1, \quad \check{l}_2 \leq 2\,\bar{t} = \bar{t}\,l_2, \quad \check{l}_3 \leq \bar{t}^{\,2} + 3\,\bar{t} - \bar{t}\,\hat{t} = \bar{t}\,l_3\ .$$

In the case $\hat{t} = \bar{t}$ one obtains

$$\check{l}_1 < \bar{t} = \bar{t}\,l_1, \quad \check{l}_2 < 2\,\bar{t} = \bar{t}\,l_2, \quad \check{l}_3 \leq \bar{t}^{\,2} + 3\,\bar{t} - \bar{t}\,\hat{t} = \bar{t}\,l_3\ .$$

Now assume that $\check{l}_{j-1} \leq \bar{t}\,l_{j-1}$ is valid for $j > 3$. If $\check{l}_j = \check{l}_{j-1} + \tilde{l}_j$ one has

$$\check{l}_j \;=\; \check{l}_{j-1} + \tilde{l}_j \leq \bar{t}\,l_{j-1} + \bar{t}(l_j - l_{j-1}) = \bar{t}\,l_j\ .$$

If $\check{l}_j = \tilde{l}_j + \hat{t} - 1$ it follows that

$$\check{l}_j = \tilde{l}_j + \hat{t} - 1 \leq \bar{t}\,l_j - \tilde{l}_j + \hat{t} - 1 < \bar{t}\,l_j\ .$$

Therefore the last asserted inequality is proven, which completes the proof. ∎

Now everything is prepared to construct the auxiliary parallel reversal schedules to reverse $\tilde{l}_k = \bar{t}\,l_{k-2} - \hat{t} + 1$ physical steps with up to k resources each of which has the property of processor-checkpoint convertibility.

Theorem B.1 (Auxiliary Reversal Schedules for $\bar{t} > 2$ and $\hat{t} < \bar{t}$). *Suppose the given uniform one-step evolution F determines the temporal complexities $\bar{t} > 2$ and $\hat{t} < \bar{t}$ to perform one reverse step and one recording step, respectively. Define the sequences l_k, $\tilde{l}_k$, and the corresponding $\check{l}_j$ as in Lemma B.1 for $k \in \mathbb{N}$. Then for $k \geq 2$ there exist feasible parallel schedules $\tilde{S}_k$ for the reversal of $\tilde{l}_k$ physical steps such that the resource profile $R(\tilde{S}_k)$ fulfils for $k \geq 3$*

$$\begin{array}{llll}
\tilde{c}_k^j = 0, & \tilde{p}_k^j = 1, & \tilde{s}_k^j = 1 & 1 \leq j \leq \bar{t} \\
 & \tilde{p}_k^j \leq \lceil \frac{i+1}{2} \rceil, & \tilde{s}_k^j \leq i & \bar{t}\, l_{i-1} < j \leq \bar{t}\, l_i,\ 2 \leq i \leq k \\
 & \tilde{p}_k^j \leq \lceil \frac{k+2}{2} \rceil, & \tilde{s}_k^j \leq k+1 & \bar{t}\, l_k < j \leq \bar{t}\, \tilde{l}_k \\
\tilde{c}_k^j = k-1, & \tilde{p}_k^j = 1, & \tilde{s}_k^j = k & \bar{t}\, \tilde{l}_k < j \leq \bar{t}\, \tilde{l}_k + \check{l}_1 \\
\tilde{c}_k^j = k-i, & \tilde{p}_k^j = 1, & \tilde{s}_k^j = k-i+1 & \begin{cases} \bar{t}\, \tilde{l}_k + \check{l}_{i-1} < j \leq \bar{t}\, \tilde{l}_k + \check{l}_i \\ 2 \leq i \leq k-1\,. \end{cases}
\end{array} \tag{B.45}$$

Proof. By definition one has $\check{l}_{k-1} = \tilde{l}_k + \hat{t} - 1$ for $k \geq 3$ because $j = k-1$ ensures that $k + j = 2k - 1$ is odd. Therefore one obtains $\bar{t}\, \tilde{l}_k + \check{l}_{k-1} = (\bar{t}+1)\tilde{l}_k + \hat{t} - 1$ and the resource profile (B.45) is well defined.

In order to prove the assertion appropriate feasible parallel schedules $\tilde{S}_k$ reversing $\tilde{l}_k$ physical steps are constructed and analyzed for $k \geq 2$. For these $\tilde{S}_k$ the resource profiles $R(\tilde{S}_k)$ fulfil Equation (B.45) if $k \geq 3$.

For $k = 2$ consider $\tilde{S}_2$ to reverse $\tilde{l}_2 = \bar{t} - \hat{t} + 1$ physical steps as illustrated in Fig. 4.35 for $\bar{t} = 4$ and $\hat{t} = 2$. I.e. no checkpoint writing is performed except that one copying the initial state 0. One obtains for each physical step $i \in \{0, \ldots, \bar{t} - \hat{t} - 1\}$ that $\hat{t} + i < \bar{t}$. Therefore during each reverse step $\bar{F}_{i+1}$ only one processor is needed to perform a forward sweep from the initial state 0 to state i and to perform the recording step $\hat{F}_i$. The corresponding resource profile of $\tilde{S}_2$ is given by

$$\begin{array}{llll}
\tilde{c}_2^j = 0, \ \tilde{p}_2^j = 1, \ \tilde{s}_2^j = 1 & \text{for} & 1 \leq j \leq \bar{t} \\
\tilde{c}_2^j = 0, \ \tilde{p}_2^j = 2, \ \tilde{s}_2^j = 2 & \text{for} & \bar{t} < j \leq \bar{t} + \hat{t} \\
\tilde{c}_2^j = 1, \ \tilde{p}_2^j = 1, \ \tilde{s}_2^j = 2 & \text{for} & \bar{t} + \hat{t} < j \leq 2\bar{t} \\
\tilde{c}_2^j = 1, \ \tilde{p}_2^j \leq 2, \ \tilde{s}_2^j \leq 3 & \text{for} & 2\,\bar{t} < j \leq \bar{t}\, \tilde{l}_2 \\
\tilde{c}_2^j = 1, \ \tilde{p}_2^i = 1, \ \tilde{s}_2^i = 2 & \text{for} & \bar{t}\, \tilde{l}_2 < j \leq (\bar{t}+1)\tilde{l}_2 + \hat{t} - 1\,.
\end{array} \tag{B.46}$$

Furthermore, the schedule $\tilde{S}_3$ provides a possibility to reverse $\tilde{l}_3 = 2\bar{t} - \hat{t} + 1$ physical steps in minimal time using two checkpoints that store the initial state 0 and state $\tilde{l}_3 - 2$ as shown in Fig. 4.35 for the case $\bar{t} = 4$ and $\hat{t} = 2$. For $1 \leq j \leq \bar{t}(\bar{t} - \hat{t} + 1)$, i.e. the physical steps i with $i = 0, \ldots, \bar{t} - \hat{t} - 1$, the resource profile of $\tilde{S}_3$ is identical to the one of $\tilde{S}_2$. For the physical step $i = \bar{t} - \hat{t}$ the equation $\bar{t} = \hat{t} + \bar{t} - \hat{t}$ is valid. Therefore during the execution of $\bar{F}_{i+1}$ and $\bar{F}_{i+2}$ again only one processor is needed to perform the necessary actions in order to start $\bar{F}_i$ and $\bar{F}_{i+1}$ in time, respectively. This yields $\tilde{p}_3^j \leq 2$ for the computational cycles $\bar{t}\, \tilde{l}_2 < j \leq \bar{t}(\bar{t} - \hat{t} + 3) = \bar{t}\, l_3$ taking the processor executing the reverse steps into account. For the physical steps $i = \bar{t} - \hat{t} + 1, \ldots, \tilde{l}_3 - 2$ one has

$$\hat{t} + i \leq \hat{t} + 2\,\bar{t} - \hat{t} + 1 - 2 = 2\,\bar{t} - 1\,. \tag{B.47}$$

Hence for these physical steps at most two processors are needed to perform the corresponding forward sweeps and recording steps $\hat{F}_i$. This yields

$$\tilde{p}_3^j \leq 3 \quad \text{and} \quad \tilde{s}_3^j \leq 4 \quad \text{for} \quad \bar{t}\, l_3 < j \leq \bar{t}(\tilde{l}_3 - 1)$$

because of the additional processor performing the reverse steps $\bar{F}_i$. During the reverse step $\bar{F}_{\tilde{l}_3-1}$ in the computational cycles $\bar{t}(\tilde{l}_3 - 1) < j \leq \bar{t}(\tilde{l}_3 - 1) + \hat{t}$ there are the checkpoint storing the initial state 0, perhaps the processor performing the forward sweep belonging to the recording step $\hat{F}_{\tilde{l}_3-3}$, and the processor evaluating $\hat{F}_{\tilde{l}_3-2}$. Therefore one has

$$\tilde{p}_3^j \leq 3 \quad \text{and} \quad \tilde{s}_3^j \leq 4 \quad \text{for} \quad \bar{t}(\tilde{l}_3 - 1) < j \leq \bar{t}(\tilde{l}_3 - 1) + \hat{t}\ .$$

The processor performing the recording step $\hat{F}_{\tilde{l}_3-2}$ is not needed anymore. Moreover from now on there is a checkpoint storing state $\tilde{l}_3 - 2$. It follows that

$$\tilde{p}_3^j \leq 2 \quad \text{and} \quad \tilde{s}_3^j \leq 4 \quad \text{for} \quad \bar{t}(\tilde{l}_3 - 1) + \hat{t} < j \leq \bar{t}\, \tilde{l}_3\ .$$

Subsequently one processor performs the recording step $\hat{F}_{\tilde{l}_3-1}$ and one physical step after the checkpoint writing that copies state $\tilde{l}_3 - 2$. The forward sweep belonging to $\hat{F}_{\tilde{l}_3-3}$ is started in a computational cycle j with $j < \bar{t}\, \tilde{l}_3$ because of Inequality (B.47). This yields

$$\tilde{c}_3^j = 2, \quad \tilde{p}_3^j = 1, \quad \tilde{s}_3^j = 3 \quad \text{for} \quad \bar{t}\, \tilde{l}_3 < j \leq \bar{t}\, \tilde{l}_3 + \check{l}_1\ .$$

For the remaining computational cycles one obtains $\tilde{c}_3^j = 1$, $\tilde{p}_3^j = 1$, as well as $\tilde{s}_3^j = 2$ because the checkpoint writing that copies state $\tilde{l}_3 - 2$ is performed in the computational cycle $\tilde{l}_3 + \hat{t} + 1$. Summarizing the above one obtains the resource profile

$$\begin{array}{llllll}
\tilde{c}_3^j = 0, & \tilde{p}_3^j = 1, & \tilde{s}_3^j = 1 & \text{for} & 1 \leq j \leq \bar{t} \\
\tilde{c}_3^j = 0, & \tilde{p}_3^j = 2, & \tilde{s}_3^j = 2 & \text{for} & \bar{t} < j \leq \bar{t} + \hat{t} \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j = 1, & \tilde{s}_3^j = 2 & \text{for} & \bar{t} + \hat{t} < j \leq 2\bar{t} \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j \leq 2, & \tilde{s}_3^j \leq 3 & \text{for} & 2\bar{t} < j \leq \bar{t} l_3 \\
\tilde{c}_3^j \leq 2, & \tilde{p}_3^j \leq 3, & \tilde{s}_3^j \leq 4 & \text{for} & \bar{t} l_3 < j \leq \bar{t} \tilde{l}_3 \\
\tilde{c}_3^j = 2, & \tilde{p}_3^j = 1, & \tilde{s}_3^j = 3 & \text{for} & \bar{t} \tilde{l}_3 < j \leq \bar{t} \tilde{l}_3 + \hat{t} + 1 \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j = 1, & \tilde{s}_3^j = 2 & \text{for} & \bar{t} \tilde{l}_3 + \hat{t} + 1 < j \leq (\bar{t} + 1)\tilde{l}_3 + \hat{t} - 1\ .
\end{array} \tag{B.48}$$

Now it has been proven that $\tilde{S}_3$ fulfils (B.45).

An induction on k will be used to show that for $k > 3$ the resource profile $R(\tilde{S}_k)$ of the parallel reversal schedule $\tilde{S}_k$ constructed as depicted in Fig 4.36 fulfils Equation (B.45). First, this will be proven for $\tilde{S}_4$. Then it will be shown that $\tilde{S}_k$ fulfils (B.45) for arbitrary $k > 4$, if (B.45) holds for $\tilde{S}_{k-1}$ and $\tilde{S}_{k-2}$.

In order to construct $\tilde{S}_4$ the parallel reversal schedule $\tilde{S}_3$ is applied for the first computational cycles. Hence one has

$$\begin{array}{llll}
\tilde{c}_4^j = \tilde{c}_3^j = 0, & \tilde{p}_4^j = \tilde{p}_3^j = 1, & \tilde{s}_4^j = \tilde{s}_3^j = 1 & 1 \leq j \leq \bar{t} \\
& \tilde{p}_4^j = \tilde{p}_3^j \leq \lceil \frac{i+1}{2} \rceil, & \tilde{s}_4^j = \tilde{s}_3^j \leq i & \bar{t}\, l_{i-1} < j \leq \bar{t}\, l_i, i = 2, 3 \\
& \tilde{p}_4^j = \tilde{p}_3^j \leq 3, & \tilde{s}_4^j = \tilde{s}_3^j \leq 4 & \bar{t}\, l_3 < j \leq \bar{t}\, \tilde{l}_3\ .
\end{array} \tag{B.49}$$

One parallel reversal schedule $\tilde{S}_2$ is placed at the vertex of $\tilde{S}_3$. One finds for $\bar{t}\,\tilde{l}_3 < j \leq (\bar{t}+1)\tilde{l}_3 + \hat{t} - 1$

$$\tilde{p}_4^j \;=\; \tilde{p}_3^j + \tilde{p}_2^{j-\bar{t}\,\tilde{l}_3} \leq 3 \quad \text{and} \quad \tilde{s}_4^j \;=\; \tilde{s}_3^j + \tilde{s}_2^{j-\bar{t}\,\tilde{l}_3} \leq 4 \tag{B.50}$$

because $2\,\bar{t} = \tilde{l}_3 + \hat{t} - 1 \leq \bar{t}^{\,2} - \bar{t}\,\hat{t} + \bar{t} = \bar{t}\,\tilde{l}_2$. Now $\tilde{S}_3$ is completed and one checkpoint stores the initial state 0. It follows for the computational cycles $(\bar{t}+1)\tilde{l}_3 + \hat{t} - 1 < j \leq \bar{t}(\tilde{l}_3 + \tilde{l}_2)$ that

$$\tilde{p}_4^j \;=\; \tilde{p}_2^{j-\bar{t}\,\tilde{l}_3} \leq 2 \quad \text{and} \quad \tilde{s}_4^j \;=\; 1 + \tilde{s}_2^{j-\bar{t}\,\tilde{l}_3} \leq 4\,. \tag{B.51}$$

The vertex of the first parallel reversal schedule $\tilde{S}_2$ is reached and therefore the second one starts. Placing $\tilde{S}_2$ at the vertex of $\tilde{S}_2$ yields

$$\tilde{p}_4^j = \tilde{p}_2^{j-\bar{t}\,\tilde{l}_3} + \tilde{p}_2^{j-\bar{t}(\tilde{l}_3+\tilde{l}_2)} \leq 2 \;\text{ and }\; \tilde{s}_4^j \;= 1 + \tilde{s}_2^{j-\bar{t}\,\tilde{l}_3} + \tilde{s}_2^{j-\bar{t}(\tilde{l}_3+\tilde{l}_2)} \leq 4 \tag{B.52}$$

for $\bar{t}(\tilde{l}_3+\tilde{l}_2) < j \leq \bar{t}(\tilde{l}_3+\tilde{l}_2)+\bar{t}$ taking the checkpoint that stores the initial state 0 into account. The equality $\bar{t}(\tilde{l}_3 + \tilde{l}_2) + \bar{t} = \bar{t}\,\tilde{l}_3 + (\bar{t}+1)\tilde{l}_2 + \hat{t} - 1$ ensures that the first parallel reversal schedule $\tilde{S}_2$ is completed. From now on one processor performs the forward sweep from the initial state 0 to state $\tilde{l}_3 - 1$ to start the evaluation of the first $\tilde{S}_2$ in time. Therefore one obtains

$$\tilde{p}_4^j = 1 + \tilde{p}_2^{j-\bar{t}(\tilde{l}_3+\tilde{l}_2)} \leq 3 \quad \text{and} \quad \tilde{s}_4^j = 2 + \tilde{s}_2^{j-\bar{t}(\tilde{l}_3+\tilde{l}_2)} \leq 4\,. \tag{B.53}$$

for the computational cycles $\bar{t}(\tilde{l}_3+\tilde{l}_2)+\bar{t} < j \leq \bar{t}(\tilde{l}_3+\tilde{l}_2)+2\,\bar{t} = \bar{t}\,l_4$. Combining Equations (B.49) up to (B.53) one finds that

$$\tilde{p}_4^j \leq 3 \quad \text{and} \quad \tilde{s}_4^j \leq 4 \quad \text{for} \quad \bar{t}\,l_3 < j \leq \bar{t}\,l_4\,. \tag{B.54}$$

Going on until the vertex of the second parallel reversal schedule $\tilde{S}_2$ is reached for $\bar{t}\,l_4 < j \leq \bar{t}(\tilde{l}_3 + 2\,\tilde{l}_2)$ follows

$$\tilde{p}_4^j = 1 + \tilde{p}_2^{j-\bar{t}(\tilde{l}_3+\tilde{l}_2)} \leq 3 \quad \text{and} \quad \tilde{s}_4^j = 2 + \tilde{s}_2^{j-\bar{t}(\tilde{l}_3+\tilde{l}_2)} \leq 5 \tag{B.55}$$

taking the additional processor and the checkpoint that stores the initial state 0 into account. Now $\bar{t}-2$ parallel reversal schedules $\tilde{S}_2$ are applied to construct $\tilde{S}_4$. Considering only these parallel reversal schedules and the checkpoint that stores the initial state 0 and forgetting for a moment the additional forward sweeps that are necessary, for the non-overlapping regions in accordance to Equation (B.46) and for the overlapping ones in accordance to Equation (B.52) one obtains

$$\tilde{p}_4^j \leq 2 \quad \text{and} \quad \tilde{s}_4^j \leq 4\,.$$

In order to perform the ith $\tilde{S}_2$, $i = 1, \ldots, \bar{t}-1$, in time the corresponding forward sweep from the initial state 0 to state $\tilde{l}_3 + (i-1)\tilde{l}_2$ would start in the computational cycle $j = (\bar{t}+1)(\tilde{l}_3 + i\tilde{l}_2) + \hat{t} - 1$ and would end in the computational cycle $j = \bar{t}(\tilde{l}_3 + i\tilde{l}_2) + \bar{t} + 1$. Furthermore, it follows that

$$(\bar{t}+1)(\tilde{l}_3 + i\tilde{l}_2) + \hat{t} - 1 \leq \bar{t}(\tilde{l}_3 + (i+1)\tilde{l}_2 + 2)\,.$$

Using $(\bar{t}+1)(\tilde{l}_3+\tilde{l}_2)+\hat{t}-1 \leq \bar{t}(\tilde{l}_3+2\,\tilde{l}_2+1)$ it is possible to define the upper bound i_u and the lower bound i_l as

$$\begin{aligned} i_u &\equiv \max_{1\leq i\leq \bar{t}-1} \left\{ (\bar{t}+1)(\tilde{l}_3+i\tilde{l}_2)+\hat{t}-1 \leq \bar{t}(\tilde{l}_3+(i+1)\tilde{l}_2+1) \right\} \\ i_l &\equiv \min\{i \mid i = \bar{t}-2m > i_u \text{ and } m \in \mathbb{N}\} \ . \end{aligned}$$

Then one has for all i with $i \leq i_u$ that only one processor is needed for the forward sweeps. Now for $i = \bar{t}-2m$ with $m \in \mathbb{N}$ and $i \geq i_l$ a forward sweep from the initial state 0 to state $\tilde{l}_3+(i-1)\tilde{l}_2$ is performed in time. During these forward sweeps a checkpoint writing that copies state $\tilde{l}_3+(i-2)\tilde{l}_2$ is performed. This construction procedure is depicted in Fig. B.6. It follows that in addition

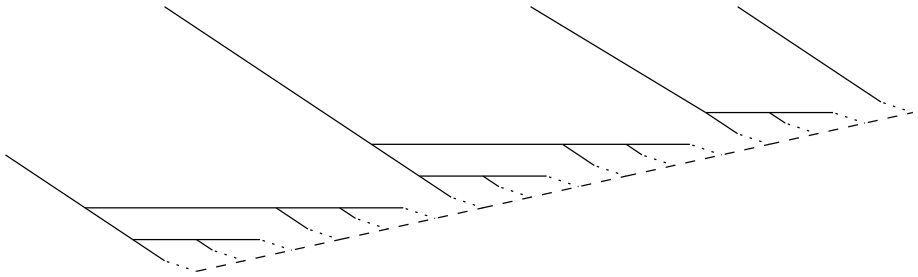


Figure B.6: Placing of Processors and Checkpoints

to the checkpoint storing the initial state 0 one processor is needed during the computational cycles $\bar{t}(\tilde{l}_3+i\tilde{l}_2+1) < j \leq (\bar{t}+1)(\tilde{l}_3+i\tilde{l}_2)+\hat{t}-1$ and one additional checkpoint for $\bar{t}(\tilde{l}_3+(i-1)\tilde{l}_2+1) < j \leq \bar{t}(\tilde{l}_3+i\tilde{l}_2)+2\,\bar{t}-\hat{t}+1$. This yields

$$\tilde{p}_4^j \leq 3, \quad \tilde{s}_4^j \leq 5 \quad \text{for} \quad \bar{t}(\tilde{l}_3+(i+1)\tilde{l}_2+1) < j \leq \bar{t}(\tilde{l}_3+(i+1)\tilde{l}_2+2) \tag{B.56}$$

because here at most one processor and one checkpoint are needed in addition to the processors and checkpoints required by two $\tilde{S}_2$ and the checkpoint storing the initial state. Furthermore, one has

$$\tilde{p}_4^j \leq 3, \quad \tilde{s}_4^j \leq 5 \quad \text{for} \quad \bar{t}(\tilde{l}_3+i\tilde{l}_2+2) < j \leq \bar{t}(\tilde{l}_3+(i+1)\tilde{l}_2+1) \tag{B.57}$$

because here at most one processor is needed in addition to the processors and checkpoints required by two $\tilde{S}_2$ and the checkpoint storing the initial state. Moreover it follows that

$$\tilde{p}_4^j \leq 3, \quad \tilde{s}_4^j \leq 5 \quad \text{for} \quad \bar{t}(\tilde{l}_3+i\tilde{l}_2+1) < j \leq \bar{t}(\tilde{l}_3+i\tilde{l}_2+2) \tag{B.58}$$

because here at most one processor and one checkpoint are needed in addition to the processors and checkpoints required by two $\tilde{S}_2$ and the checkpoint storing the initial state. Finally one obtains

$$\tilde{p}_4^j \leq 2, \quad \tilde{s}_4^j \leq 5 \quad \text{for} \quad \bar{t}(\tilde{l}_3+(i-1)\tilde{l}_2+2) < j \leq \bar{t}(\tilde{l}_3+i\tilde{l}_2+1) \tag{B.59}$$

because here at most one checkpoint is needed in addition to the processors and checkpoints required by two $\tilde{S}_2$ and the checkpoint storing the initial state.

For all $i < i_l - 1$ one has as mentioned above that only one processor is needed for the evaluation of the forward sweeps to state $\tilde{l}_3 + (i-1)\tilde{l}_2$. Therefore the inequalities

$$\tilde{p}_4^j \leq 3 \quad \text{and} \quad \tilde{s}_4^j \leq 5 \tag{B.60}$$

are valid for $\bar{t}(\tilde{l}_3 + 2\,\tilde{l}_2) < j \leq \bar{t}(\tilde{l}_3 + i_l\tilde{l}_2) + \bar{t}$. One obtains from Equations (B.55) up to (B.60) that

$$\tilde{p}_4^j \leq 3 \quad \text{and} \quad \tilde{s}_4^j \leq 5 \quad \text{for} \quad \bar{t}\,l_4 < j \leq \bar{t}\,\tilde{l}_4\,. \tag{B.61}$$

Now the vertex of the $\bar{t}$th parallel reversal schedule is reached. Taking the checkpoints that store the initial state 0 and state $\tilde{l}_3 + (\bar{t}-2)\tilde{l}_2$ into account one obtains for $\bar{t}\,\tilde{l}_4 < j \leq \bar{t}\,\tilde{l}_4 + \bar{t}$

$$\begin{aligned} &\tilde{c}_4^j = 2 + \tilde{c}_2^{j-\bar{t}(\tilde{l}_4-\tilde{l}_2)} = 3, \quad \tilde{p}_4^j = \tilde{p}_2^{j-\bar{t}(\tilde{l}_4-\tilde{l}_2)} = 1, \\ &\tilde{s}_4^j = 2 + \tilde{s}_2^{j-\bar{t}(\tilde{l}_4-\tilde{l}_2)} = 4\,. \end{aligned} \tag{B.62}$$

The final parallel reversal $\tilde{S}_2$ is completed and from now on one processor performs a forward sweep from the initial state 0 to state $\tilde{l}_3 + (\bar{t}-1)\tilde{l}_2$. The checkpoint writing that copies state $\tilde{l}_3 + (\bar{t}-2)\tilde{l}_2$ is performed in the computational cycle $\bar{t}\,\tilde{l}_4 + \bar{t} + \tilde{l}_2 = \bar{t}\,\tilde{l}_4 + \check{l}_2$. This yields

$$\tilde{c}_4^j = 2, \quad \tilde{p}_4^j = 1, \quad \tilde{s}_4^j = 3 \quad \text{for} \quad \bar{t}\,\tilde{l}_4 + \bar{t} < j \leq \bar{t}\,\tilde{l}_4 + \check{l}_2 \tag{B.63}$$

as well as

$$\tilde{c}_4^j = 1, \quad \tilde{p}_4^j = 1, \quad \tilde{s}_4^j = 2 \quad \text{for} \quad \bar{t}\,\tilde{l}_4 + \check{l}_2 < j \leq \bar{t}\,\tilde{l}_4 + \check{l}_3\,. \tag{B.64}$$

The Equations (B.49), (B.54), and (B.61) up to (B.64) show that the resource profile $R(\tilde{S}_4)$ fulfils Equation (B.45).

From the results proven so far follows that $\tilde{S}_3$ and $\tilde{S}_4$ fulfil Equation (B.45). Now it will be shown that this equation holds for the resource profile of $\tilde{S}_k$ with $k > 4$ if $\tilde{S}_k$ is constructed as depicted in Fig. 4.36 and explained below using $\tilde{S}_{k-1}$ and $\tilde{S}_{k-2}$, which fulfil (B.45), too.

Once more the first computational cycles of $\tilde{S}_k$ are formed by $\tilde{S}_{k-1}$. Hence one finds

$$\begin{aligned} \tilde{c}_k^j = \tilde{c}_{k-1}^j = 0,\ \tilde{p}_k^j = \tilde{p}_{k-1}^j = 1, \qquad & \tilde{s}_k^j = \tilde{s}_{k-1}^j = 1 \quad 1 \leq j \leq \bar{t} \\ \tilde{p}_k^j = \tilde{p}_{k-1}^j \leq \lceil \tfrac{i+1}{2} \rceil,\ & \tilde{s}_k^j = \tilde{s}_{k-1}^j \leq i \quad \begin{cases} \bar{t}\,l_{i-1} < j \leq \bar{t}\,l_i, \\ 2 \leq i \leq k-1 \end{cases} \\ \tilde{p}_k^j = \tilde{p}_{k-1}^j \leq \lceil \tfrac{k+1}{2} \rceil,\ & \tilde{s}_k^j = \tilde{s}_{k-1}^j \leq k \quad \bar{t}\,l_{k-1} < j \leq \bar{t}\,\tilde{l}_{k-1} \end{aligned} \tag{B.65}$$

and the vertex of $\tilde{S}_{k-1}$ is reached. Here the first parallel reversal schedule $\tilde{S}_{k-2}$ is placed. As shown in Lemma B.1 the relation $\check{l}_j \leq \bar{t}l_j$ is valid. Therefore one has

$$\begin{aligned} &\tilde{p}_k^j = \tilde{p}_{k-1}^j + \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \leq 1 + \left\lceil \frac{i+1}{2} \right\rceil = \left\lceil \frac{i+3}{2} \right\rceil \leq \left\lceil \frac{k+1}{2} \right\rceil, \\ &\tilde{s}_k^j = \tilde{s}_{k-1}^j + \tilde{s}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \leq k - i + i = k \end{aligned} \tag{B.66}$$

for $\bar{t}(\tilde{l}_{k-1} + l_{i-1}) < j \leq \bar{t}(\tilde{l}_{k-1} + l_i)$ and $i = 1, \ldots, k-2$. If $\tilde{S}_{k-1}$ is completed earlier the values of $\tilde{p}_k^j$ and $\tilde{s}_k^j$ are overestimated by Equation (B.66). Nevertheless, it suffices to prove the assertion. For the next computational cycles the checkpoint storing the initial state 0 must be taken into account. This yields

$$\tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq \left\lceil \frac{k+1}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j = 1 + \tilde{s}_{k-2}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq k \tag{B.67}$$

for $\bar{t}(\tilde{l}_{k-1} + l_{k-2}) < j \leq \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2})$. The vertex of the first parallel reversal schedule $\tilde{S}_{k-2}$ is reached and the second one starts yielding

$$\begin{aligned} \tilde{p}_k^j &= \tilde{p}_{k-2}^{j-\bar{t}\,\tilde{l}_{k-1}} + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 1 + \left\lceil \frac{i+1}{2} \right\rceil = \left\lceil \frac{i+3}{2} \right\rceil \leq \left\lceil \frac{k}{2} \right\rceil, \\ \tilde{s}_k^j &= \tilde{s}_{k-2}^{j-\bar{t}\,\tilde{l}_{k-1}} + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 1 + k - 1 - i + i = k \end{aligned} \tag{B.68}$$

for $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + \tilde{l}_{i-1}) < j \leq \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + \tilde{l}_i)$ and $i = 1, \ldots, k-3$. Now it is certain that the first parallel reversal schedule $\tilde{S}_{k-2}$ is completed. Hence there exists a processor performing the forward sweep from the initial state to state $\tilde{l}_{k-1}$ in addition to the checkpoint storing the initial state 0. It follows that

$$\begin{aligned} \tilde{p}_k^j &\leq 1 + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 1 + \left\lceil \frac{k-1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil, \\ \tilde{s}_k^j &\leq 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 2 + k - 2 = k \end{aligned} \tag{B.69}$$

for $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + \tilde{l}_{k-3}) < j \leq \bar{t}(\tilde{l}_{k-2} + \tilde{l}_{k-2} + l_{k-2})$. Because the identity $\tilde{l}_{k-2} + \tilde{l}_{k-2} + l_{k-2} = l_k$ is valid as shown in Lemma B.1 Equations (B.66) up to (B.69) prove that

$$\tilde{p}_k^j \leq \left\lceil \frac{k+1}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq k \quad \text{for} \quad \bar{t}\,\tilde{l}_{k-1} < j \leq \bar{t}\,l_k\,. \tag{B.70}$$

Going on until the vertex of the second parallel schedule $\tilde{S}_{k-2}$ is reached one obtains for $\bar{t}\,l_k < j \leq \bar{t}(\tilde{l}_{k-1} + 2\tilde{l}_{k-2})$

$$\begin{aligned} \tilde{p}_k^j &\leq 1 + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 1 + \left\lceil \frac{k}{2} \right\rceil = \left\lceil \frac{k+2}{2} \right\rceil, \\ \tilde{s}_k^j &\leq 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 2 + k - 1 = k + 1\,. \end{aligned} \tag{B.71}$$

Now $\bar{t} - 2$ parallel reversal schedules $\tilde{S}_{k-2}$ are applied to construct $\tilde{S}_k$. Considering only these parallel reversal schedules and the checkpoint storing the initial state 0 and forgetting for a moment the additional forward sweeps that are necessary, for the non-overlapping regions in accordance to Equation (B.45) and for the overlapping ones in accordance to Equation (B.68) follows

$$\tilde{p}_k^j \leq \left\lceil \frac{k}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq k\,.$$

In order to perform the ith $\tilde{S}_{k-2}$, $i = 1, \ldots, \bar{t} - 1$, in time the corresponding forward sweep from the initial state 0 to state $\tilde{l}_{k-1} + (i-1)\tilde{l}_{k-2}$ would start in

the computational cycle $j = (\bar{t}+1)(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \hat{t} - 1$ and would end in the computational cycle $j = \bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \check{l}_{k-3} + 1$. Furthermore, one has

$$(\bar{t}+1)(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \hat{t} - 1 \leq \bar{t}(\tilde{l}_{k-1} + (i+1)\tilde{l}_{k-2} + l_{k-2}) \; .$$

Using $(\bar{t}+1)(\tilde{l}_{k-1} + \tilde{l}_{k-2}) + \hat{t} - 1 \leq \bar{t}(\tilde{l}_{k-1} + 2\tilde{l}_{k-2}) + \check{l}_{k-3}$ it is possible to define the upper bound i_u and the lower bound i_l as

$$\begin{aligned} i_u &\equiv \max_{1 \leq i \leq \bar{t}-1} \left\{ (\bar{t}+1)(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \hat{t} - 1 \leq \bar{t}(\tilde{l}_{k-1} + (i+1)\tilde{l}_{k-2}) + \check{l}_{k-3} \right\} \\ i_l &\equiv \min \{ i \mid i = \bar{t} - 2m > i_u \text{ and } m \in \mathbb{N} \} \; . \end{aligned} \tag{B.72}$$

Then one has for all i with $i \leq i_u$ that only one processor is needed for the forward sweeps. Now for all $i = \bar{t} - 2m$ with $m \in \mathbb{N}$ and $i \geq i_l$ a forward sweep from the initial state 0 to state $\tilde{l}_{k-1} + (i-1)\tilde{l}_{k-2}$ is performed in time. During these forward sweeps a checkpoint writing that copies state $\tilde{l}_{k-1} + (i-2)\tilde{l}_{k-2}$ is performed. This construction procedure is the same as that one depicted already in Fig. 4.36. Therefore it follows that in addition to the checkpoint storing the initial state 0 one processor is needed for the computational cycles $\bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \check{l}_{k-3} < j \leq (\bar{t}+1)(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \hat{t} - 1$. Moreover during the computational cycles $\bar{t}(\tilde{l}_{k-1} + (i-1)\tilde{l}_{k-2}) + \check{l}_{k-3} < j \leq \bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + 2\,\tilde{l}_{k-2}$ one additional checkpoint is needed. This yields

$$\tilde{p}_k^j \leq 1 + \left\lceil \frac{k}{2} \right\rceil = \left\lceil \frac{k+2}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq 3 + k - 2 = k + 1 \tag{B.73}$$

for $\bar{t}(\tilde{l}_{k-1} + (i+1)\tilde{l}_{k-2}) + \check{l}_{k-3} < j \leq \bar{t}(\tilde{l}_{k-1} + (i+1)\tilde{l}_{k-2} + l_{k-2})$ because in these computational cycles at most one processor and one checkpoint are needed in addition to the processors and checkpoints required by two $\tilde{S}_{k-2}$ and the checkpoint storing the initial state. Furthermore, one has

$$\tilde{p}_k^j \leq 1 + \left\lceil \frac{k}{2} \right\rceil = \left\lceil \frac{k+2}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq 2 + k - 1 = k + 1 \tag{B.74}$$

for $\bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + 2\,\tilde{l}_{k-2} < j \leq \bar{t}(\tilde{l}_{k-1} + (i+1)\tilde{l}_{k-2}) + \check{l}_{k-3}$ because in these computational cycles at most one processor is needed in addition to the processors and checkpoints required by two $\tilde{S}_{k-2}$ and the checkpoint storing the initial state. Moreover it follows that

$$\tilde{p}_k^j \leq 1 + \left\lceil \frac{k}{2} \right\rceil = \left\lceil \frac{k+2}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq 3 + k - 2 = k + 1 \tag{B.75}$$

for $\bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \check{l}_{k-3} < j \leq \bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + 2\,\tilde{l}_{k-2}$ because in these computational cycles at most one processor and one checkpoint are needed in addition to the processors and checkpoints required by two $\tilde{S}_{k-2}$ and the checkpoint storing the initial state. Finally, one obtains

$$\tilde{p}_k^j \leq \left\lceil \frac{k}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq 2 + k - 1 = k + 1 \tag{B.76}$$

for $\bar{t}(\tilde{l}_{k-1} + (i-1)\tilde{l}_{k-2} + l_{k-2}) < j \leq \bar{t}(\tilde{l}_{k-1} + i\tilde{l}_{k-2}) + \check{l}_{k-3}$ because in these computational cycles at most one checkpoint is needed in addition to the processors and checkpoints required by two $\tilde{S}_{k-2}$ and the checkpoint storing the

initial state. For all $i < i_l - 1$ only one processor is necessary for the evaluation of the forward sweeps from the initial state 0 to state $\tilde{l}_{k-1} + (i-1)\tilde{l}_{k-2}$ because of Equation (B.72). Therefore the inequalities

$$\tilde{p}_k^j \leq \left\lceil \frac{k+2}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq k+1 \tag{B.77}$$

are valid for $\bar{t}(\tilde{l}_{k-1} + 2\,\tilde{l}_{k-2}) < j \leq \bar{t}(\tilde{l}_{k-1} + (i_l - 1)\,\tilde{l}_{k-2} + l_{k-2})$. Combining Equation (B.71) and Equations (B.73) up to (B.77) one obtains for the number of processors used and the number of resources needed in the computational cycles $\bar{t}\,l_k < j \leq \bar{t}\,\tilde{l}_k$ the upper bounds

$$\tilde{p}_k^j \leq \left\lceil \frac{k+2}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq k+1\,. \tag{B.78}$$

Now the vertex of the $\bar{t}$th parallel reversal schedule $\tilde{S}_{k-2}$ is reached. Taking the checkpoints that store the initial state 0 and state $\tilde{l}_{k-1} + (\bar{t}-2)\tilde{l}_{k-2}$ into account it follows

$$\begin{aligned} \tilde{c}_k^j &= 2 + \tilde{c}_{k-2}^{j-\bar{t}(\tilde{l}_k - \tilde{l}_{k-2})} = k - i, \qquad \tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_k - \tilde{l}_{k-2})} = 1, \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_k - \tilde{l}_{k-2})} = k - i + 1 \end{aligned} \tag{B.79}$$

for the computational cycles $\bar{t}\,\tilde{l}_k + \check{l}_{i-1} < j \leq \bar{t}\,\tilde{l}_k + \check{l}_i$ and $i = 1, \ldots, k-3$. Then the final parallel reversal schedule $\tilde{S}_{k-2}$ is completed. From now on one processor performs a forward sweep that advances from the initial state 0 to state $\tilde{l}_{k-1} + (\bar{t}-1)\tilde{l}_{k-2}$. Furthermore a checkpoint writing copies the state $\tilde{l}_{k-1} + (\bar{t}-2)\tilde{l}_{k-2}$ in the computational cycle $\bar{t}\,\tilde{l}_k + \check{l}_{k-3} + \tilde{l}_{k-2}$. Hence one finds that

$$\tilde{c}_k^j = 2, \quad \tilde{p}_k^j = 1, \quad \tilde{s}_k^j = 3 \quad \text{for} \quad \bar{t}\,\tilde{l}_k + \check{l}_{k-3} < j \leq \bar{t}\,\tilde{l}_k + \check{l}_{k-2} \tag{B.80}$$

with $\check{l}_{k-3} + \tilde{l}_{k-2} = \check{l}_{k-2}$ as well as

$$\tilde{c}_k^j = 1, \quad \tilde{p}_k^j = 1, \quad \tilde{s}_k^j = 2 \quad \text{for} \quad \bar{t}\,\tilde{l}_k + \check{l}_{k-2} < j \leq \bar{t}\,\tilde{l}_k + \check{l}_{k-1}\,. \tag{B.81}$$

Combining Equations (B.65), (B.70), (B.78), (B.79), and (B.80) one finds that Equation (B.45) holds for the resource profile $R(\tilde{S}_k)$ of $\tilde{S}_k$, which completes the proof. ∎

Suppose that the reverse steps and recording steps of the one-step evolution F under consideration determine the same temporal complexities $\bar{t} = \hat{t} > 2$. Then a construction principle quite similar to the one used in the last proof is applied for the auxiliary parallel reversal schedules $\tilde{S}_k$ for the reversal of $\tilde{l}_k = \bar{t}\,l_{k-1} - \hat{t} + 1$ physical steps.

Theorem B.2 (Auxiliary Reversal Schedules for $\hat{t} = \bar{t} > 2$).
Suppose the given uniform one-step evolution F determines the temporal complexities $\bar{t} = \hat{t} > 2$ to perform one reverse step and one recording step, respectively. Define the sequences l_k, $\tilde{l}_k$, and the corresponding $\check{l}_j$ as in Lemma B.1 for $k \in \mathbb{N}$. Then for $k \geq 3$ there exist feasible parallel schedules $\tilde{S}_k$ for the reversal of $\tilde{l}_k$ physical steps such that the resource profiles $R(\tilde{S}_k)$ fulfil for $k \geq 4$

$$\begin{array}{llll}
\tilde{c}_k^j = 0, & \tilde{p}_k^j = 1, & \tilde{s}_k^j = 1 & 1 \leq j \leq \bar{t} \\
 & \tilde{p}_k^j \leq \lceil \frac{i+1}{2} \rceil, & \tilde{s}_k^j \leq i & \bar{t}\, l_{i-1} < j \leq \bar{t}\, l_i,\ 2 \leq i \leq k \\
 & \tilde{p}_k^j \leq \lceil \frac{k+2}{2} \rceil, & \tilde{s}_k^j \leq k{+}1 & \bar{t}\, l_k < j \leq \bar{t}\, \tilde{l}_k \\
\tilde{c}_k^j = k{-}2{-}o, & \tilde{p}_k^j = 2{+}o, & \tilde{s}_k^j = k & \bar{t}\, \tilde{l}_k < j \leq \bar{t}\, \tilde{l}_k{+}\check{l}_1 \\
\tilde{c}_k^j = k{-}2{-}o, & \tilde{p}_k^j = 1{+}o, & \tilde{s}_k^j = k{-}1 & \bar{t}\, \tilde{l}_k{+}\check{l}_1 < j \leq \bar{t}\, \tilde{l}_k{+}\check{l}_2 \\
\tilde{c}_k^j = k{-}i, & \tilde{p}_k^j = 1, & \tilde{s}_k^j = k{-}i{+}1 & \begin{cases} \bar{t}\, \tilde{l}_k{+}\check{l}_{i-1} < j \leq \bar{t}\, \tilde{l}_k{+}\check{l}_i \\ i = 3, \ldots, k-1 \end{cases}
\end{array} \tag{B.82}$$

with $o \equiv \lceil \frac{k}{2} \rceil - \lfloor \frac{k}{2} \rfloor$.

Proof. As shown in Lemma B.1 the identity $\check{l}_{k-1} = \tilde{l}_k + \bar{t} - 1$ is valid for $k \geq 4$. Therefore the resource profile (B.82) is well defined. In order to prove the assertion feasible parallel schedules $\tilde{S}_k$ reversing $\tilde{l}_k$ physical steps are constructed and analyzed for $k \geq 3$. For these parallel reversal schedules the resource profile fulfil Equation (B.82) if $k \geq 4$.

The parallel schedule $\tilde{S}_3$ to reverse $\tilde{l}_3 = \bar{t}{+}1$ physical steps is constructed as depicted in Fig. B.7 for the case $\bar{t} = \hat{t} = 3$, i.e. without any checkpoint writing besides the one copying the initial state. Therefore the corresponding resource

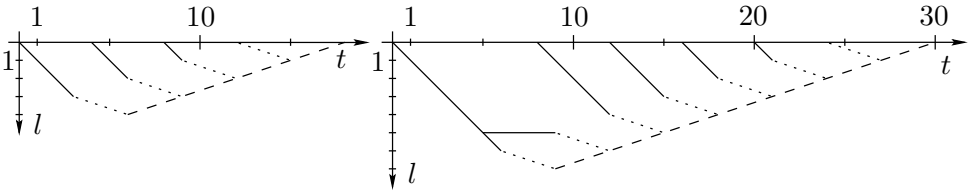


Figure B.7: Parallel Reversal Schedules $\tilde{S}_3$ and $\tilde{S}_4$ for $\bar{t} = \hat{t} = 3$

profile is clear for the computational cycles $1 \leq j \leq 3\,\bar{t}$. For the physical steps $i \in \{0, \ldots, \bar{t}\,\}$ one needs $\bar{t}{+}i \leq 2\,\bar{t}$ computational cycles to perform the recording step $\hat{F}_i$ and the corresponding forward sweep from the initial state 0 to state i. Therefore two processors suffice to execute the recording steps and the forward sweeps for these physical steps during the computational cycles $3\,\bar{t} < j \leq \bar{t}\,\tilde{l}_3$. One needs $2\,\bar{t} - 1$ computational cycles to perform $\hat{F}_{\bar{t}-1}$ and the appropriate forward sweep. Moreover there are the checkpoint storing the initial state and one processor performing $\hat{F}_{\bar{t}}$ during the forward sweep belonging to $\hat{F}_{\bar{t}-1}$. This yields

$$\tilde{c}_3^j = 1, \quad \tilde{p}_3^j = 2, \quad \text{and} \quad \tilde{s}_3^j = 3 \quad \text{for} \quad \bar{t}\,\tilde{l}_3 < j \leq \bar{t}\,\tilde{l}_3 + \bar{t} - 1\,.$$

For the remaining computational cycles one obtains

$$\tilde{c}_3^j = 1, \quad \tilde{p}_3^j = 1, \quad \text{and} \quad \tilde{s}_3^j = 2 \quad \text{with} \quad \bar{t}\,\tilde{l}_3 + \bar{t} - 1 < j \leq \bar{t}\,\tilde{l}_3 + 2\,\bar{t}\,.$$

In detail the resource profile is given by

$$\begin{array}{llll}
\tilde{c}_3^j = 0, & \tilde{p}_3^j = 1, & \tilde{s}_3^j = 1 & \text{for} \quad 1 \le j \le \bar{t} \\
\tilde{c}_3^j = 0, & \tilde{p}_3^j = 2, & \tilde{s}_3^j = 2 & \text{for} \quad \bar{t} < j \le 2\,\bar{t} \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j = 2, & \tilde{s}_3^j = 3 & \text{for} \quad 2\,\bar{t} < j \le 3\,\bar{t} \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j \le 3, & \tilde{s}_3^j \le 4 & \text{for} \quad 3\,\bar{t} < j \le \bar{t}\,\tilde{l}_3 \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j = 2, & \tilde{s}_3^j = 3 & \text{for} \quad \bar{t}\,\tilde{l}_3 < j \le \bar{t}\,\tilde{l}_3 + \bar{t} - 1 \\
\tilde{c}_3^j = 1, & \tilde{p}_3^j = 1, & \tilde{s}_3^j = 2 & \text{for} \quad \bar{t}\,\tilde{l}_3 + \bar{t} - 1 < j \le \bar{t}\,\tilde{l}_3 + 2\,\bar{t}\,.
\end{array} \tag{B.83}$$

The construction of the parallel schedule $\tilde{S}_4$ to reverse $\tilde{l}_4 = 2\,\bar{t}+1$ physical steps is depicted also in Fig. B.7. This yields immediately

$$\begin{array}{llll}
\tilde{c}_4^j = 0, & \tilde{p}_4^j = 1, & \tilde{s}_4^j = 1 & \text{for} \quad 1 \le j \le \bar{t} \\
\tilde{c}_4^j = 0, & \tilde{p}_4^j = 2, & \tilde{s}_4^j = 2 & \text{for} \quad \bar{t} < j \le 2\,\bar{t} \\
\tilde{c}_4^j = 1, & \tilde{p}_4^j = 2, & \tilde{s}_4^j = 3 & \text{for} \quad 2\,\bar{t} < j \le 3\,\bar{t}\,.
\end{array} \tag{B.84}$$

Furthermore, for each physical step i with $2 \le i \le \bar{t}$ the corresponding recording step and the corresponding forward sweeps are performed by one processor. As explained above one has

$$\tilde{p}_4^j \le 3 \quad \text{and} \quad \tilde{s}_4^j \le 4 \quad \text{for} \quad 3\,\bar{t} < j \le \bar{t}(\bar{t}+2)\,.$$

Now for all $i = 2\,\bar{t} - 2m$ with $m \in \mathbb{N}$ and $i \ge \bar{t}+1$ a forward sweep is performed to start the corresponding recording step in time. During these forward sweeps a checkpoint writing that copies state $i-1$ is performed if $i > \bar{t}+1$. Hence if $\bar{t}$ is odd one processor performs a forward sweep from the initial state 0 to state $\bar{t}+1$, which yields

$$\tilde{p}_4^j \le 3 \quad \text{and} \quad \tilde{s}_4^j \le 4 \quad \text{for} \quad \bar{t}(\bar{t}+2) < j \le \bar{t}(\bar{t}+4) = \bar{t}\,l_4\,.$$

If $\bar{t}$ is even a checkpoint writing that copies state $\bar{t}+1$ is performed by the forward sweep to state $\bar{t}+2$. Therefore one finds

$$\tilde{p}_4^j \le 2 \quad \text{and} \quad \tilde{s}_4^j \le 4 \quad \text{for} \quad \bar{t}(\bar{t}+2) < j \le \bar{t}(\bar{t}+4) = \bar{t}\,l_4\,.$$

Hence one obtains

$$\tilde{p}_4^j \le 3 \quad \text{and} \quad \tilde{s}_4^j \le 4 \quad \text{for} \quad \bar{t}\,l_3 < j \le \bar{t}\,l_4\,. \tag{B.85}$$

For $i = 2\,\bar{t} - 2m$ with $m \in \mathbb{N}$ and $i > \bar{t}+1$ one processor is needed to perform the recording step and the corresponding forward sweep during the computational cycles $\bar{t}(i+1) < j \le \bar{t}(i+2) + i$ as well as one checkpoint during the computational cycles $\bar{t}(i+1) < j \le \bar{t}(i+2) + 1$. Because $\bar{t} + i \le 3\,\bar{t}$ no more than two processors are required in one of these computational cycles to perform the recording steps and the forward sweeps needed. Moreover no more than one checkpoint is necessary in addition to the one storing the initial state. This yields

$$\tilde{p}_4^j \le 3 \quad \text{and} \quad \tilde{s}_4^j \le 5 \quad \text{for} \quad \bar{t}\,l_4 < j \le \bar{t}(2\,\bar{t}+1) = \bar{t}\,\tilde{l}_4\,. \tag{B.86}$$

Then there are two checkpoints storing the initial state and state $\tilde{l}_4 - 2$, respectively. The checkpoint writing that copies state $\tilde{l}_4 - 2$ is executed in the computational cycle $\bar{t}\,\tilde{l}_4+\bar{t}+1$. Furthermore, two processors perform the recording step $\hat{F}_{\tilde{l}_4-1}$ and the forward sweep belonging to $\hat{F}_{\tilde{l}_4-3}$, respectively. Therefore it follows that

$$\tilde{c}_4^j = 2, \quad \tilde{p}_4^j = 2, \quad \text{and} \quad \tilde{s}_4^j = 4 \quad \text{for} \quad \bar{t}\,\tilde{l}_4 < j \leq \bar{t}\,\tilde{l}_4 + \bar{t} - 2\,. \tag{B.87}$$

For the remaining computational cycles one obtains

$$\tilde{c}_4^j = 2, \quad \tilde{p}_4^j = 1, \quad \text{and} \quad \tilde{s}_4^j = 3 \quad \text{with} \quad \bar{t}\,\tilde{l}_4 + \bar{t} - 2 < j \leq \bar{t}\,\tilde{l}_4 + \bar{t} + 1 \tag{B.88}$$

as well as

$$\tilde{c}_4^j = 1, \quad \tilde{p}_4^j = 1, \quad \text{and} \quad \tilde{s}_4^j = 2 \quad \text{with} \quad \bar{t}\,\tilde{l}_4 + \bar{t} + 1 < j \leq \bar{t}\,\tilde{l}_4 + \check{l}_3\,. \tag{B.89}$$

Equations (B.84) and (B.85) up to (B.89) show that $\tilde{S}_4$ fulfils Equation (B.82).

For $k \geq 5$ the parallel reversal schedules are constructed as depicted earlier in Fig. 4.36. It will be proven that the Equation (B.82) holds for the resource profile of $\tilde{S}_5$. Then it will be shown that this equation is fulfilled by $R(\tilde{S}_k)$ with $k \geq 6$ if $\tilde{S}_k$ is constructed according to the same scheme using $\tilde{S}_{k-1}$ and $\tilde{S}_{k-2}$, which fulfil (B.82) also.

In order to construct $\tilde{S}_5$ the parallel reversal schedule $\tilde{S}_4$ is applied for the first computational cycles. This yields

$$\begin{array}{llll} \tilde{c}_5^j = 0, \quad \tilde{p}_5^j = 1, & \tilde{s}_5^j = 1 & \text{for} & 1 \leq j \leq \bar{t} \\ \tilde{p}_5^j \leq \lceil \frac{i+1}{2} \rceil, & \tilde{s}_5^j \leq i & \text{for} & \bar{t}\,l_{i-1} < j \leq \bar{t}\,l_i,\ i = 2,3,4 \\ \tilde{p}_5^j \leq \lceil \frac{k+2}{2} \rceil, & \tilde{s}_5^j \leq k+1 & \text{for} & \bar{t}\,l_4 < j \leq \bar{t}\,\tilde{l}_4\,. \end{array} \tag{B.90}$$

Now the first parallel reversal schedule $\tilde{S}_3$ is placed at the vertex of $\tilde{S}_4$. It follows that

$$\tilde{p}_5^j = \tilde{p}_4^j + \tilde{p}_3^{j-\bar{t}\,\tilde{l}_4} \leq 3, \quad \tilde{s}_5^j = \tilde{s}_4^j + \tilde{s}_3^{j-\bar{t}\,\tilde{l}_4} \leq 5 \quad \text{for} \quad \bar{t}\,\tilde{l}_4 < j \leq \bar{t}(\tilde{l}_4 + 3) \tag{B.91}$$

and the parallel reversal schedule $\tilde{S}_4$ is completed. From now on a checkpoint storing the initial state must be taken into account. Hence one obtains

$$\tilde{p}_5^j = \tilde{p}_3^{j-\bar{t}\,\tilde{l}_4} \leq 3, \quad \tilde{s}_5^j = 1 + \tilde{s}_3^{j-\bar{t}\,\tilde{l}_4} \leq 5 \quad \text{for} \quad \bar{t}(\tilde{l}_4+3) < j \leq \bar{t}(\tilde{l}_4+\tilde{l}_3) \tag{B.92}$$

and the vertex of the first $\tilde{S}_3$ is reached. The second parallel reversal schedule $\tilde{S}_3$ starts. Therefore one has

$$\tilde{p}_5^j = \tilde{p}_3^{j-\bar{t}\,\tilde{l}_4} + \tilde{p}_3^{j-\bar{t}(\tilde{l}_4+\tilde{l}_3)} \leq 3 \quad \text{and} \quad \tilde{s}_5^j = 1 + \tilde{s}_3^{j-\bar{t}\tilde{l}_4} + \tilde{s}_3^{j-\bar{t}(\tilde{l}_4+\tilde{l}_3)} \leq 5 \tag{B.93}$$

for $\bar{t}(\tilde{l}_4+\tilde{l}_3) < j \leq \bar{t}(\tilde{l}_4+\tilde{l}_3+2)$ taking also the checkpoint that stores the initial state into account. Then the first parallel schedule $\tilde{S}_3$ is completed. From now on one processor performs a forward sweep from the initial state 0 to state $\tilde{l}_4$. It follows that

$$\tilde{p}_5^j = 1 + \tilde{p}_3^{j-\bar{t}(\tilde{l}_4+\tilde{l}_3)} \leq 3 \quad \text{and} \quad \tilde{s}_5^j = 2 + \tilde{s}_3^{j-\bar{t}(\tilde{l}_4+\tilde{l}_3)} \leq 5 \tag{B.94}$$

for $\bar{t}(\tilde{l}_4 + \tilde{l}_3 + 2) < j \leq \bar{t}(\tilde{l}_4 + \tilde{l}_3 + 3)$. Equations (B.91) up to (B.94) show that

$$\tilde{p}_5^j \leq 3 \quad \text{and} \quad \tilde{s}_5^j \leq 5 \quad \text{for} \quad \bar{t}\, l_4 < j \leq \bar{t}\, l_5 \tag{B.95}$$

because $\tilde{l}_4 + \tilde{l}_3 + 3 = 3\,\bar{t} + 5 = l_5$. Going on until the vertex of the second parallel reversal schedule $\tilde{S}_3$ is reached one obtains

$$\tilde{p}_5^j = 1 + \tilde{p}_3^{j-\bar{t}(\tilde{l}_4+\tilde{l}_3)} \leq 4 \quad \text{and} \quad \tilde{s}_5^j = 2 + \tilde{s}_3^{j-\bar{t}(\tilde{l}_4+\tilde{l}_3)} \leq 6 \tag{B.96}$$

for $\bar{t}\, l_5 < j \leq \bar{t}(\tilde{l}_4 + 2\,\tilde{l}_3)$. Now $\bar{t} - 2$ parallel reversal schedules $\tilde{S}_3$ are applied to construct $\tilde{S}_5$. Considering only these parallel reversal schedules and the checkpoint that stores the initial state 0 and forgetting for a moment the additional forward sweeps that are necessary, for the non-overlapping regions in accordance to Equation (B.83) and for the overlapping ones in accordance to Equation (B.93) follows

$$\tilde{p}_5^j \leq 3 \quad \text{and} \quad \tilde{s}_5^j \leq 5\,. \tag{B.97}$$

In order to perform the ith $\tilde{S}_3$, $i = 1, \ldots, \bar{t} - 1$, in time the corresponding forward sweep from the initial state 0 to state $\tilde{l}_3 + (i-1)\tilde{l}_2$ would start in the computational cycle $j = (\bar{t}+1)(\tilde{l}_4 + i\tilde{l}_3) + \bar{t} - 1$ and would end in the computational cycle $j = \bar{t}(\tilde{l}_4 + i\tilde{l}_3) + 2\,\bar{t} + 1$. Furthermore, it follows that

$$(\bar{t}+1)(\tilde{l}_4 + i\tilde{l}_3) + \bar{t} - 1 \leq \bar{t}(\tilde{l}_4 + (i+1)\tilde{l}_3) + 2\,\bar{t}\,.$$

Because of Inequalities (B.97) one has

$$\tilde{p}_5^j \leq 4 \quad \text{and} \quad \tilde{s}_5^j \leq 6 \quad \text{for} \quad \bar{t}\, l_5 < j \leq \bar{t}(\tilde{l}_4 + \bar{t}\,\tilde{l}_3) = \bar{t}\,\tilde{l}_5\,. \tag{B.98}$$

For $i = \bar{t}-1$ the forward sweep comprises $\tilde{l}_4 + (\bar{t}-2)\tilde{l}_3 = \bar{t}(\bar{t}+1) - 1$ computational cycles. This yields the equalities

$$\begin{array}{lll}
\tilde{p}_5^j = 1 + \tilde{p}_3^{j-\bar{t}(\tilde{l}_5-\tilde{l}_3)} = 3, & \tilde{s}_5^j = 2 + \tilde{s}_3^{j-\bar{t}(\tilde{l}_5-\tilde{l}_3)} = 5 & \text{for } 0 < j - \bar{t}\,\tilde{l}_5 \leq \check{l}_1 \\
\tilde{p}_5^j = 1 + \tilde{p}_3^{j-\bar{t}(\tilde{l}_5-\tilde{l}_3)} = 2, & \tilde{s}_5^j = 2 + \tilde{s}_3^{j-\bar{t}(\tilde{l}_5-\tilde{l}_3)} = 4 & \text{for } \check{l}_1 < j - \bar{t}\,\tilde{l}_5 \leq \check{l}_2 \\
\tilde{p}_5^j = \tilde{p}_3^{j-\bar{t}(\tilde{l}_5-\tilde{l}_3)} = 1, & \tilde{s}_5^j = 1 + \tilde{s}_3^{j-\bar{t}(\tilde{l}_5-\tilde{l}_3)} = 3 & \text{for } \check{l}_2 < j - \bar{t}\,\tilde{l}_5 \leq \check{l}_3\,.
\end{array} \tag{B.99}$$

Now the $\bar{t}$th parallel reversal schedule $\tilde{S}_3$ is finished. Therefore it follows for the remainder of $\tilde{S}_5$ that

$$\tilde{c}_5^j = 1, \quad \tilde{p}_5^j = 1, \quad \text{and} \quad \tilde{s}_5^j = 2 \quad \text{for} \quad \bar{t}\,\tilde{l}_5 + \check{l}_3 < j \leq \bar{t}\,\tilde{l}_5 + \check{l}_4\,. \tag{B.100}$$

Equations (B.90), (B.95), and (B.98) up to (B.100) show that Equation (B.82) holds for the resource profile $R(\tilde{S}_5)$.

From the results achieved so far follows that $R(\tilde{S}_4)$ and $R(\tilde{S}_5)$ fulfil Equation (B.82). Now it will be shown that this equation holds for the resource profile $R(\tilde{S}_k)$ with $k > 5$ if $\tilde{S}_k$ is constructed as depicted already in Fig. 4.36 and explained below using $\tilde{S}_{k-1}$ and $\tilde{S}_{k-2}$ the resource profiles of which fulfil Equation (B.82).

Once more the first computational cycles of $\tilde{S}_k$ are formed by the first computational cycles of $\tilde{S}_{k-1}$. This yields

$$\begin{aligned} \tilde{c}_k^j = \tilde{c}_{k-1}^j = 0,\ \tilde{p}_k^j = \tilde{p}_{k-1}^j = 1, \qquad \tilde{s}_k^j = \tilde{s}_{k-1}^j = 1 \quad & 1 \le j \le \bar{t} \\ \tilde{p}_k^j = \tilde{p}_{k-1}^j \le \lceil \tfrac{i+1}{2} \rceil,\ \tilde{s}_k^j = \tilde{s}_{k-1}^j \le i \quad & \begin{cases} \bar{t}\, l_{i-1} < j \le \bar{t}\, l_i, \\ 2 \le i \le k-1 \end{cases} \\ \tilde{p}_k^j = \tilde{p}_{k-1}^j \le \lceil \tfrac{k+1}{2} \rceil,\ \tilde{s}_k^j = \tilde{s}_{k-1}^j \le k \quad & \bar{t}\, l_{k-1} < j \le \bar{t}\, \tilde{l}_{k-1}. \end{aligned} \tag{B.101}$$

Now the first parallel reversal schedule $\tilde{S}_{k-2}$ is placed at the vertex of $\tilde{S}_{k-1}$. As shown in Lemma B.1 the relation $\check{l}_j \le \bar{t}\, l_j$ is valid. Therefore one has

$$\begin{aligned} \tilde{p}_k^j = \tilde{p}_{k-1}^j + \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \le 3{+}1 = 4 \le \lceil \tfrac{k+1}{2} \rceil \qquad & 0 < j - \bar{t}\, \tilde{l}_{k-1} \le \bar{t} \\ \tilde{p}_k^j = \tilde{p}_{k-1}^j + \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \le 2{+}2 = 4 \le \lceil \tfrac{k+1}{2} \rceil \qquad & \bar{t} < j - \bar{t}\, \tilde{l}_{k-1} \le 2\,\bar{t} \end{aligned} \tag{B.102}$$

and for $\bar{t}(\tilde{l}_{k-1} + l_{i-1}) < j \le \bar{t}(\tilde{l}_{k-1} + l_i)$ with $i = 3, \ldots, k-2$

$$\tilde{p}_k^j = \tilde{p}_{k-1}^j + \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \le 1{+}\left\lceil \frac{i+1}{2} \right\rceil \le \left\lceil \frac{k+1}{2} \right\rceil . \tag{B.103}$$

For the sum $\tilde{s}_k^j$ of $\tilde{c}_k^j$ and $\tilde{p}_k^j$ follows that

$$\tilde{s}_k^j = \tilde{s}_{k-1}^j + \tilde{s}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \le k - i + i = k \tag{B.104}$$

for $\bar{t}(\tilde{l}_{k-1}{+}l_{i-1}) < j \le \bar{t}(\tilde{l}_{k-1}{+}l_i)$ with $i = 1, \ldots, k{-}2$. Then $\tilde{S}_{k-1}$ is completed. From now on the checkpoint storing the initial state must be taken into account. This yields

$$\tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \le \left\lceil \frac{k+1}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j = 1 + \tilde{s}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} \le k \tag{B.105}$$

for $\bar{t}(\tilde{l}_{k-1} + l_{k-2}) < j \le \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2})$. Now the vertex of the first parallel reversal schedule $\tilde{S}_{k-2}$ is reached and the second one is started. Therefore one has

$$\tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \le 3 + 1 = 4 \le \left\lceil \frac{k+1}{2} \right\rceil \tag{B.106}$$

for the computational cycles $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2}) < j \le \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + 1)$ as well as

$$\tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \le 2 + 2 = 4 \le \left\lceil \frac{k+1}{2} \right\rceil \tag{B.107}$$

for $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + 1) < j \le \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + 2)$. For the computational cycles $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + l_{i-1}) < j \le \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + l_i)$ with $i = 3, \ldots, k-3$ one obtains

$$\tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \le 1{+}\left\lceil \frac{i+1}{2} \right\rceil \le \left\lceil \frac{k+1}{2} \right\rceil . \tag{B.108}$$

For the sum $\tilde{s}_k^j$ of $\tilde{c}_k^j$ and $\tilde{p}_k^j$ follows that

$$\tilde{s}_k^j = 1 + \tilde{s}_{k-2}^{j-\bar{t}\tilde{l}_{k-1}} + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \le 1 + k - i - 1 + i = k \tag{B.109}$$

for $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + l_{i-1}) < j \leq \bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + l_i)$ with $i = 1, \ldots, k-3$. Then the first parallel reversal schedule $\tilde{S}_{k-2}$ is completed. Now there exists one processor performing the forward sweep from the initial state 0 to state $\tilde{l}_{k-1}$ in addition to the checkpoint storing the initial state. Hence it follows that

$$\begin{aligned} \tilde{p}_k^j &= 1 + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 1 + \left\lceil \frac{k-1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil , \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 2 + k - 2 = k \end{aligned} \tag{B.110}$$

for $\bar{t}(\tilde{l}_{k-1} + \tilde{l}_{k-2} + l_{k-3}) < j \leq \bar{t}(\tilde{l}_{k-2} + \tilde{l}_{k-2} + l_{k-2})$. Because the identity $\tilde{l}_{k-2} + \tilde{l}_{k-2} + l_{k-2} = l_k$ is valid as shown in Lemma B.1 Equations (B.102) up to (B.110) prove that

$$\tilde{p}_k^j \leq \left\lceil \frac{k+1}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq k \quad \text{for} \quad \bar{t}\,\tilde{l}_{k-1} < j \leq \bar{t}\,l_k \, . \tag{B.111}$$

Going on until the vertex of the second parallel schedule $\tilde{S}_{k-2}$ is reached it follows that

$$\begin{aligned} \tilde{p}_k^j &\leq 1 + \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 1 + \left\lceil \frac{k}{2} \right\rceil = \left\lceil \frac{k+2}{2} \right\rceil, \\ \tilde{s}_k^j &\leq 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_{k-1}+\tilde{l}_{k-2})} \leq 2 + k - 1 = k + 1 \end{aligned} \tag{B.112}$$

for $\bar{t}\,l_k < j \leq \bar{t}(\tilde{l}_{k-1} + 2\,\tilde{l}_{k-2})$. Now $\bar{t} - 2$ parallel reversal schedules $\tilde{S}_{k-2}$ are applied to construct $\tilde{S}_k$. Using exactly the same argumentation as in Theorem B.1 one can show that

$$\tilde{p}_k^j \leq \left\lceil \frac{k+2}{2} \right\rceil \quad \text{and} \quad \tilde{s}_k^j \leq k+1 \quad \text{for} \quad \bar{t}\,l_k < j \leq \bar{t}\,\tilde{l}_k \, . \tag{B.113}$$

The vertex of the $\bar{t}$th parallel reversal schedule $\tilde{S}_{k-2}$ is reached. Taking the checkpoints that store the initial state 0 and state $\tilde{l}_{k-1} + (\bar{t}-2)\tilde{l}_{k-2}$ into account it follows for $\bar{t}\,\tilde{l}_k < j \leq \bar{t}\,\tilde{l}_k + \check{l}_1$ if k is even that

$$\begin{aligned} \tilde{c}_k^j &= 2 + \tilde{c}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 4 = k - 2, \quad \tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2, \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 2 = k \end{aligned} \tag{B.114}$$

and if k is odd that

$$\begin{aligned} \tilde{c}_k^j &= 2 + \tilde{c}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 5 = k - 3, \quad \tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 3, \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 2 = k \, . \end{aligned} \tag{B.115}$$

Moreover for $\bar{t}\,\tilde{l}_k + \check{l}_1 < j \leq \bar{t}\,\tilde{l}_k + \check{l}_2$ in the case that k is even one obtains

$$\begin{aligned} \tilde{c}_k^j &= 2 + \tilde{c}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 4 = k - 2, \quad \tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 1, \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 3 = k - 1 \end{aligned} \tag{B.116}$$

and in the case that k is odd

$$\begin{aligned}\tilde{c}_k^j &= 2 + \tilde{c}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 5 = k - 3, \quad \tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2, \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 2 + k - 3 = k - 1 \, .\end{aligned} \tag{B.117}$$

For $\bar{t}\,\tilde{l}_k + \check{l}_{i-1} < j \leq \bar{t}\,\tilde{l}_k + \check{l}_i$ and $i = 3, \ldots, k-3$ one finds

$$\begin{aligned}\tilde{c}_k^j &= 2 + \tilde{c}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = k - i, \quad \tilde{p}_k^j = \tilde{p}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = 1, \\ \tilde{s}_k^j &= 2 + \tilde{s}_{k-2}^{j-\bar{t}(\tilde{l}_k-\tilde{l}_{k-2})} = k - i + 1 \, .\end{aligned} \tag{B.118}$$

The $\bar{t}$th parallel reversal schedule $\tilde{S}_{k-2}$ is completed. From now on one processor performs a forward sweep from the initial state 0 to state $\tilde{l}_{k-1} + (\bar{t}-1)\tilde{l}_{k-2}$. Furthermore, a checkpoint writing that copies state $\tilde{l}_{k-1} + (\bar{t}-2)\tilde{l}_{k-2}$ is performed in the computational cycle $\bar{t}\,\tilde{l}_k + \check{l}_{k-3} + \tilde{l}_{k-2}$. This yields

$$\tilde{c}_k^j = 2, \quad \tilde{p}_k^j = 1, \quad \text{and} \quad \tilde{s}_k^j = 3 \quad \text{for} \quad \bar{t}\,\tilde{l}_k + \check{l}_{k-3} < j \leq \bar{t}\,\tilde{l}_k + \check{l}_{k-2} \tag{B.119}$$

with $\check{l}_{k-3} + \tilde{l}_{k-2} = \check{l}_{k-2}$ as well as

$$\tilde{c}_k^j = 1, \quad \tilde{p}_k^j = 1, \quad \text{and} \quad \tilde{s}_k^j = 2 \quad \text{for} \quad \bar{t}\,\tilde{l}_k + \check{l}_{k-2} < j \leq \bar{t}\,\tilde{l}_k + \check{l}_{k-1} \, . \tag{B.120}$$

Combining Equations (B.101), (B.111), and (B.113) up to (B.120) it is shown that Equation (B.82) holds for $R(\tilde{S}_k)$, which completes the proof. ∎

The auxiliary feasible reversal schedule $\tilde{S}_k$ developed in the Theorems B.1 and B.2 are applied to construct the desired feasible parallel reversal schedules S_k for the temporal complexities $\bar{t}$ and $\hat{t}$ with $\bar{t} > 2$ and $\hat{t} \leq \bar{t}$:

Proof of Theorem 4.10, Page 96:
Obviously, one has $2 < 2 + \hat{t}/\bar{t} \leq 3$ because $\hat{t} \leq \bar{t}$. Therefore the construction of appropriate parallel reversal schedules S_k for $k \leq 2$ is clear. They are displayed by Fig. B.8 for $\bar{t} = 4$ and $\hat{t} = 2$. These parallel reversal schedules for $k \in \{1, 2\}$ need no more than k processors. Hence the assertion is true for $k \leq 2$.

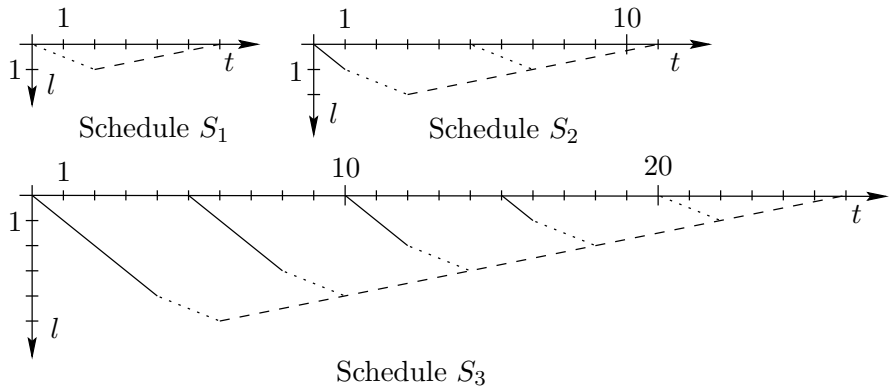


Figure B.8: Parallel Reversal Schedules S_1, S_2, and S_3 for $\bar{t} = 4$ and $\hat{t} = 2$

For $k = 3$ one obtains $l_3 = 2+\bar{t}-\hat{t}+1 = \bar{t}-\hat{t}+3$. Consider S_3 constructed as depicted in Fig. B.8. For each $i \in \{0, \ldots, \bar{t}-\hat{t}\}$ one has that $\hat{t}+i \leq \bar{t}$. Therefore during each reverse step $\bar{F}_{i+1}$ only one processor is needed to perform a forward sweep from the initial state 0 to state i and to perform the recording step $\hat{F}_i$. For $i = \bar{t}-\hat{t}$ the equation $\bar{t}+1 = \hat{t}+i+1$ is valid. Hence during the evaluation of the last computational cycle of $\bar{F}_{i+2}$ and the evaluation of $\bar{F}_{i+1}$ again one processor is needed to perform the necessary actions to start $\bar{F}_i$ in time. Hence one has $p_k^j = 2$ for the computational cycles $j = (\bar{t}-1)l_3+1, \ldots, \bar{t}\,l_3+1$. In the remaining computational cycles of S_3 one processor performs the forward sweep from state 0 to state l_3-1, the recording step $\hat{F}_{l_3-1}$ as well as the first $\bar{t}-1$ computational cycles of $\bar{F}_{l_3-1}$. Furthermore, one checkpoint stores the initial state 0. Therefore one obtains the resource profile

$$\begin{array}{llll}
c_3^j = 0, & p_3^j = 1, & s_3^j = 1 & \text{for} \quad 1 \leq j \leq \bar{t} \\
c_3^j = 0, & p_3^j = 2, & s_3^j = 2 & \text{for} \quad \bar{t} < j \leq \bar{t}+\hat{t} \\
c_3^j = 1, & p_3^j = 1, & s_3^j = 2 & \text{for} \quad \bar{t}+\hat{t} < j \leq 2\,\bar{t} \\
c_3^j = 1, & p_3^j \leq 2, & s_3^j \leq 3 & \text{for} \quad 2\,\bar{t} < j \leq \bar{t}\,l_3 \\
c_3^j = 1, & p_3^j = 2, & s_3^j = 3 & \text{for} \quad j = \bar{t}\,l_3+1 \\
c_3^j = 1, & p_3^j = 1, & s_3^j = 2 & \text{for} \quad \bar{t}\,l_3+1 < j \leq (\bar{t}+1)l_3+\hat{t}-1\,.
\end{array}$$

For $k \geq 4$ the construction of the parallel reversal schedule S_k is shown in Fig. 4.38, where $\tilde{S}_{k-1}$ denotes the parallel schedule to reverse $\tilde{l}_{k-1}$ physical steps as constructed in the Theorems B.1 and B.2. Using the induction principle in this proof it will be shown that Equation (4.29) holds for the resource profile $R(S_k)$ of S_k if $k \geq 4$. To do so first the parallel reversal schedule S_4 is constructed according to Fig. 4.38. It will be shown that Equation (4.29) holds for S_4. Then it will proven that this equation is also valid for S_k if S_k is constructed as depicted in Fig. 4.38 using S_{k-1} that fulfils Equation (4.29) and $\tilde{S}_{k-1}$ that fulfils Equation (B.45) and (B.82), respectively.

For the first computational cycles of S_4 the parallel reversal schedule $\tilde{S}_3$ is applied. Hence one obtains

$$\begin{array}{llll}
c_4^j = \tilde{c}_3^j = 0, & p_4^j = \tilde{p}_3^j = 1, & s_4^j = \tilde{s}_3^j = 1 & 1 \leq j \leq \bar{t} \\
& p_4^j = \tilde{p}_3^j \leq \lceil \frac{i+1}{2} \rceil, & s_4^j = \tilde{s}_3^j \leq i & \bar{t}\,l_{i-1} < j \leq \bar{t}\,l_i,\ i = 2,3 \\
& p_4^j = \tilde{p}_3^j \leq 3, & s_4^j = \tilde{s}_3^j \leq 4 & \bar{t}\,l_3 < j \leq \bar{t}\,\tilde{l}_3\,.
\end{array} \tag{B.121}$$

Now the vertex of $\tilde{S}_3$ is reached and S_3 starts. This yields

$$\left.\begin{array}{l}
p_4^j = \tilde{p}_3^j + p_3^{j-\bar{t}\,\tilde{l}_3} \leq 2+1 = 3, \\
s_4^j = \tilde{s}_3^j + s_3^{j-\bar{t}\,\tilde{l}_3} \leq 3+1 = 4
\end{array}\right\} \quad \bar{t}\,\tilde{l}_3 < j \leq \bar{t}\,\tilde{l}_3 + \bar{t}$$

$$\left.\begin{array}{l}
p_4^j = \tilde{p}_3^j + p_3^{j-\bar{t}\,\tilde{l}_3} \leq 1+2 = 3, \\
s_4^j = \tilde{s}_3^j + s_3^{j-\bar{t}\,\tilde{l}_3} = 2+2 = 4
\end{array}\right\} \quad \bar{t}\,\tilde{l}_3 + \bar{t} < j \leq \bar{t}\,\tilde{l}_3 + 2\,\bar{t} \tag{B.122}$$

and $\tilde{S}_3$ is completed. The checkpoint storing the initial state has to be taken

into account. It follows with $\tilde{l}_3 + l_3 = l_4$ that

$$\left.\begin{array}{l} p_4^j = p_3^{j-\bar{t}\,\tilde{l}_3} \leq 2, \\ s_4^j = 1+s_3^{j-\bar{t}\,\tilde{l}_3} \leq 1+3 = 4 \end{array}\right\} \quad \bar{t}\,\tilde{l}_3 + 2\,\bar{t} < j \leq \bar{t}(\tilde{l}_3 + l_3)$$

$$\left.\begin{array}{l} c_4^j = 1+c_3^{j-\bar{t}\,\tilde{l}_3} = 2,\ p_4^j = p_3^{j-\bar{t}\,\tilde{l}_3} = 2, \\ s_4^j = 1+s_3^{j-\bar{t}\,\tilde{l}_3} = 1+3 = 4 \end{array}\right\} \quad j = \bar{t}\,l_4 + 1 \tag{B.123}$$

$$\left.\begin{array}{l} c_4^j = 1+c_3^{j-\bar{t}\,\tilde{l}_3} = 2,\ p_4^j = p_3^{j-\bar{t}\,\tilde{l}_3} = 1, \\ s_4^j = 1+s_3^{j-\bar{t}\,\tilde{l}_3} = 1+2 = 3 \end{array}\right\} \quad \bar{t}\,l_4 + 1 < j \leq \bar{t}\,l_4 + l_3 + \hat{t} - 1$$

and the parallel reversal schedule S_3 is completed, too. The only task left is the forward sweep from the initial state to the state $\tilde{l}_3$. Therefore one has

$$c_4^j = 1, \quad p_4^j = 1, \quad \text{and} \quad s_4^j = 1 \quad \text{for} \quad l_3 < j - \bar{t}\,l_4 - \hat{t} + 1 \leq l_4\,. \tag{B.124}$$

Hence it is shown by Equations (B.121) up to (B.124) that the resource profile $R(S_4)$ fulfils Equation (4.29).

In order to construct S_k for $k > 4$ the parallel reversal schedule $\tilde{S}_{k-1}$ serves to form the first computational cycles of S_k. This yields

$$\begin{array}{lll} c_k^j = \tilde{c}_{k-1}^j = 0,\ p_k^j = \tilde{p}_{k-1}^j = 1, & s_k^j = \tilde{s}_{k-1}^j = 1 & 1 \leq j \leq \bar{t} \\ p_k^j = \tilde{p}_{k-1}^j \leq \lceil \frac{i+1}{2} \rceil, & s_k^j = \tilde{s}_{k-1}^j \leq i & \left\{\begin{array}{l} \bar{t}\,l_{i-1} < j \leq \bar{t}\,l_i, \\ 2 \leq i \leq k-1 \end{array}\right. \\ p_k^j = \tilde{p}_{k-1}^j \leq \lceil \frac{k+1}{2} \rceil, & s_k^j = \tilde{s}_{k-1}^j \leq k & \bar{t}\,l_{k-1} < j \leq \bar{t}\,\tilde{l}_{k-1} \end{array} \tag{B.125}$$

and the vertex of $\tilde{S}_{k-1}$ is reached. Therefore S_{k-1} starts. Because of the inequality $\check{l}_i \leq \bar{t}\,l_i$ shown in Lemma B.1 one can conclude

$$\begin{array}{l} \left.\begin{array}{l} p_k^j = \tilde{p}_{k-1}^j + p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq 2+1 \leq \lceil \frac{k+1}{2} \rceil, \\ s_k^j = \tilde{s}_{k-1}^j + s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq k-1+1 = k \end{array}\right\} \quad \bar{t}\,\tilde{l}_{k-1} < j \leq \bar{t}\,\tilde{l}_{k-1} + \bar{t} \\ \left.\begin{array}{l} p_k^j = \tilde{p}_{k-1}^j + p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq 1+2 \leq \lceil \frac{k+1}{2} \rceil, \\ s_k^j = \tilde{s}_{k-1}^j + s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} = k-1+1 = k \end{array}\right\} \quad \bar{t} < j - \bar{t}\,\tilde{l}_{k-1} \leq 2\,\bar{t} \end{array} \tag{B.126}$$

as well as

$$\begin{array}{l} p_k^j = \tilde{p}_{k-1}^j + p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq 1 + \left\lceil \dfrac{i+1}{2} \right\rceil \leq \left\lceil \dfrac{k+1}{2} \right\rceil, \\ s_k^j = \tilde{s}_{k-1}^j + s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq k - i + i = k \end{array} \tag{B.127}$$

for $\bar{t}(\tilde{l}_{k-1} + l_{i-1}) < j \leq \bar{t}(\tilde{l}_{k-1} + l_i)$ and $i = 3, \ldots, k-2$. Obviously, if $\check{l}_i < \bar{t}\,l_j$ then the Inequalities (B.127) overestimate the values of p_k^j and s_k^j. Nevertheless, it suffices to prove the assertion. From now on only the checkpoint storing the initial state must be taken into account. This yields

$$\left.\begin{array}{l} p_k^j = p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq \lceil \frac{k+1}{2} \rceil, \\ s_k^j = 1 + s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \leq 1 + k - 1 = k, \end{array}\right\} \quad \bar{t}(\tilde{l}_{k-1} + l_{k-2}) < j \leq \bar{t}\,l_k \tag{B.128}$$

because $\tilde{l}_{k-1} + l_{k-1} = l_k$. For the remainder of S_{k-1} one obtains

$$\left.\begin{array}{l} p_k^j = p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} = 2, \\ s_k^j = 1+s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \le 1+k-1 = k \end{array}\right\} \quad j = \bar{t}\,l_k + 1$$

$$\left.\begin{array}{l} p_k^j = p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} = 1, \\ s_k^j = 1+s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \le 1+k-2 = k-1 \end{array}\right\} \quad 1 < j-\bar{t}\,l_k \le l_3+\hat{t}-1 \qquad \text{(B.129)}$$

$$\left.\begin{array}{l} p_k^j = p_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} = 1, \\ s_k^j = 1+s_{k-1}^{j-\bar{t}\,\tilde{l}_{k-1}} \le k-i+2 \end{array}\right\} \quad \begin{array}{l} l_{i-1} < j-\bar{t}\,l_k-\hat{t}+1 \le l_i \\ 4 \le i \le k-1 \end{array}$$

and the parallel reversal schedule S_{k-1} is completed. The only task left is the forward sweep from the initial state to the state $\tilde{l}_{k-1}$. It follows that

$$c_k^j = 1, \quad p_k^j = 1, \quad \text{and} \quad s_k^j = 1 \quad \text{for} \quad l_{k-1} < j - \bar{t}\,l_k - \hat{t} + 1 \le l_k \quad \text{(B.130)}$$

because $l_{k-1} + \tilde{l}_{k-1} = l_k$. Equations (B.125) up to (B.130) show that S_k fulfils Equation (4.29), which completes the proof. ∎

The feasible parallel reversal schedules of the last proof form the base to construct the desired parallel reversal schedules S_k for reversing l_k physical steps for the temporal complexities $\bar{t}, \hat{t} \in \mathbb{N}$ with $\hat{t} > \bar{t} > 2$. The same construction principle as in the proofs of Theorem 4.6 is applied.

Proof of Theorem 4.11, Page 98:
First assume $1 \le k < 2 + \hat{t}/\bar{t}$. Then appropriate parallel reversal schedules S_k can be constructed according to the ones described in the beginning of the proof of Theorem 4.6. Hence for each physical step i, $0 \le i \le l_k - 1 = k - 1$ one processor performs a forward sweep from the initial state 0 to state i, one recording step $\hat{F}_i$, and one reverse step $\bar{F}_i$. Then it is possible to control all k processors in order to calculate the reversal appropriately (see Fig. B.5). It is easy to check that one obtains the following resource profile

$$\begin{array}{llll} c_k^j = 0, & p_k^j = i, & s_k^j = i & (i-1)\bar{t} < j \le i\,\bar{t},\ 1 \le i \le k-1 \\ c_k^j = 0, & p_k^j = k, & s_k^j = k & j = (k-1)\bar{t}+1 \\ c_k^j = 0, & p_k^j = k, & s_k^j = k & (k-1)\bar{t}+1 < j \le \bar{t}+\hat{t} \\ c_k^j = 1, & p_k^j = k-i, & s_k^j = k-i+1 & \left\{\begin{array}{l} (i-1)(\bar{t}+1) < j-\bar{t}-\hat{t} \le i(\bar{t}+1), \\ 1 \le i \le k-1\,. \end{array}\right. \end{array}$$

Therefore the assertion has been proven for $1 \le k < 2 + \hat{t}/\bar{t}$.

For $k \ge 2 + \hat{t}/\bar{t}$ parallel reversal schedules S_k needing no more than p_k^b processors are constructed for the reversal of l_k physical steps. Defining

$$r \;\equiv\; \hat{t} - \lfloor(\hat{t}-1)/\bar{t}\rfloor\bar{t}$$

yields $r \in \{1, \ldots \bar{t}\}$. Denote with $S_{k,r}$ the feasible parallel reversal schedules developed in the last proof for the reversal of $l_{k,r}$ physical steps for the temporal complexities $\bar{t} > 2$ and $r \le \bar{t}$ to perform one reverse step and one recording step, respectively. Using Equality (4.30) the desired feasible parallel reversal

schedules S_k that are based on the parallel reversal schedules of Theorem 4.10 will be constructed.

Consider for $k \geq 2 + \lceil \hat{t}/\bar{t} \rceil$ the feasible parallel reversal schedule $S_{\tilde{k},r}$ with $\tilde{k} \equiv k - (\hat{t}-r)/\bar{t}$ and r as above. Take the first $\bar{t}\, l_{\tilde{k},r} + r$ computational cycles of $S_{\tilde{k},r}$ as constructed in Theorem 4.10 and transform them into the first $\bar{t}\, l_{\tilde{k},r} + \hat{t}$ computational cycles of a parallel reversal schedule $\bar{S}_k$ such that the given value of $\hat{t}$ is taken into account. This corresponds to shifting the reverse sweep to the right as depicted in Fig. 4.25 for the case $k = 7$, $\hat{t} = 3$, and $\bar{t} = 1$. Everything else, especially the checkpoint writings, remain unchanged. Therefore in each computational cycle j with $1 \leq j \leq 2\, l_{\tilde{k},r}$ of $\bar{S}_k$ there are at most $(\hat{t}-r)/\bar{t}+1$ processors performing recording steps instead of at most one processor in the corresponding computational cycles of $S_{\tilde{k},r}$. This yields the resource profile

$$\begin{array}{lll} c_k^j = 0,\ p_k^j = i, & s_k^j = i & \left\{\begin{array}{l}(i-1)\bar{t} < j \leq i\bar{t},\\ 1 \leq i \leq (\hat{t}-r)/\bar{t}+1\end{array}\right. \\ c_k^j = 0,\ p_k^j = \frac{\hat{t}-r}{\bar{t}}+2, & s_k^j = \frac{\hat{t}-r}{\bar{t}}+2 & \hat{t}-r+\bar{t} < j \leq \hat{t}+r \\ \quad p_k^j \leq p_{\tilde{k},r}^{j-\hat{t}+r}+\frac{\hat{t}-r}{\bar{t}}, & s_k^j \leq s_{\tilde{k},r}^{j-\hat{t}+r}+\frac{\hat{t}-r}{\bar{t}} & \hat{t}+r < j \leq \bar{t}\, l_{\tilde{k},r}\,. \end{array} \tag{B.131}$$

Furthermore, for each $m = 1, \ldots, \hat{t}-1$ during the computational cycle j with $j = \bar{t}\, l_{\tilde{k},r} - \hat{t} + 1 + m$ of $S_{\tilde{k},r}$ two processors are needed for a recording step and a reverse step if $m - \lfloor (m-1)/\bar{t} \rfloor \bar{t} \leq r$ as well as one processor for the reverse step else. In the corresponding computational cycle $j = \bar{t}\, l_{\tilde{k},r} + m$ of $\bar{S}_k$ at most $(\hat{t}-r)/\bar{t}+1-\lceil (m-r)/\bar{t} \rceil$ processors are needed only for the recording steps. No reverse step has started yet. Therefore, the number of processors needed as well as the total number of resources required increase by $(\hat{t}-r)/\bar{t}-1-\lfloor (m-1)/\bar{t} \rfloor$ during each computational cycle $j = \bar{t}\, l_{\tilde{k},r} + m$ of $\bar{S}_k$ with $m = 1, \ldots, \hat{t}-1$. Moreover, the computational cycle $\bar{t}\, l_{\tilde{k},r} + r$ of the parallel reversal schedule $S_{\tilde{k},r}$ constructed in Theorem 4.10 corresponds exactly to the computational cycle $\bar{t}\, l_{\tilde{k},r} + \hat{t}$ of $\bar{S}_k$. Therefore one obtains the following resource profile for the next computational cycles of the parallel reversal schedule $\bar{S}_k$

$$\begin{array}{ll} \left.\begin{array}{l} p_k^j \leq p_{\tilde{k},r}^{j-\hat{t}+r}+(\hat{t}-r)/\bar{t}-1-\lfloor (m-1)/\bar{t} \rfloor, \\ s_k^j \leq s_{\tilde{k},r}^{j-\hat{t}+r}+(\hat{t}-r)/\bar{t}-1-\lfloor (m-1)/\bar{t} \rfloor \end{array}\right\} & j = \bar{t}\, l_{\tilde{k},r}+m,\ 1 \leq m \leq \hat{t}-1 \\ p_k^j = p_{\tilde{k},r}^{j-\hat{t}+r}, \qquad s_k^j = s_{\tilde{k},r}^{j-\hat{t}+r} & j = \bar{t}\, l_{\tilde{k},r} + \hat{t}\,. \end{array} \tag{B.132}$$

For the computational cycles $j = 1, \ldots, \bar{t}\, l_{\tilde{k},r}$ follows

$$\begin{aligned} p_k^j &\leq \max\left\{\frac{\hat{t}-r}{\bar{t}}+2, \max_{\hat{t}+r<j\leq \bar{t}\, l_{\tilde{k},r}}\left\{p_{\tilde{k},r}^{j-\hat{t}+r}+\frac{\hat{t}-r}{\bar{t}}\right\}\right\} \\ &\leq \max\left\{\frac{\hat{t}-r}{\bar{t}}+2, \left\lceil\frac{\tilde{k}+1}{2}\right\rceil+\frac{\hat{t}-r}{\bar{t}}\right\} \leq \left\lceil\frac{k+1}{2}\right\rceil+\left\lceil\frac{1}{2}\left\lfloor\frac{\hat{t}-1}{\bar{t}}\right\rfloor\right\rceil \end{aligned} \tag{B.133}$$

as well as

$$\begin{aligned} s_k^j &\leq \max\left\{\frac{\hat{t}-r}{\bar{t}}+2, \max_{\hat{t}+r<j\leq\bar{t}\,l_{\tilde{k},r}}\left\{s_{\tilde{k},r}^{j-\hat{t}+r}+\frac{\hat{t}-r}{\bar{t}}\right\}\right\} \\ &\leq \max\left\{\frac{\hat{t}-r}{\bar{t}}+2, k-\frac{\hat{t}-r}{\bar{t}}+\frac{\hat{t}-r}{\bar{t}}\right\} = k\,. \end{aligned} \tag{B.134}$$

Furthermore, for each $j = \bar{t}\,l_{\tilde{k},r} + m$ with $1 \leq m \leq \hat{t}-1$ one finds that

$$\begin{aligned} p_k^j &= p_{\tilde{k},r}^{j-\hat{t}+r} + \frac{\hat{t}-r}{\bar{t}} - 1 - \left\lfloor\frac{m-1}{\bar{t}}\right\rfloor \\ &\leq \left\lceil\frac{k+1}{2}\right\rceil + \left\lceil\frac{1}{2}\left\lfloor\frac{\hat{t}-1}{\bar{t}}\right\rfloor\right\rceil - 1 - \left\lfloor\frac{m-1}{\bar{t}}\right\rfloor, \\ s_k^j &= s_{\tilde{k},r}^{j-\hat{t}+r} + \frac{\hat{t}-r}{\bar{t}} - 1 - \left\lfloor\frac{m-1}{\bar{t}}\right\rfloor \leq k - 1 - \left\lfloor\frac{m-1}{\bar{t}}\right\rfloor. \end{aligned} \tag{B.135}$$

It is possible to put the first $\bar{t}\,l_{\tilde{k},r} + \hat{t}$ computational cycles of $\bar{S}_k$ and the computational cycles $j = \bar{t}l_{\tilde{k},r} + r + 1, \ldots, (\bar{t}+1)l_{\tilde{k},r} + r - 1$ of $S_{\tilde{k},r}$ together to form a complete parallel reversal schedule $\bar{S}_k$ for $l_{\tilde{k},r}$ physical steps and the given $\hat{t}$. This can be done, because the computational cycle $\bar{t}\,l_{\tilde{k},r} + \hat{t}$ of $\bar{S}_k$ is equal to the computational cycle $\bar{t}\,l_{\tilde{k},r} + r$ of $S_{\tilde{k},r}$. The resulting parallel reversal schedule corresponds to the one shown in Fig. 4.26 for $k = 7$, $\hat{t} = 3$, and $\bar{t} = 1$. For the computational cycles $j = \bar{t}\,l_{\tilde{k},r} + \hat{t} + 1, \ldots, (\bar{t}+1)l_{\tilde{k},r} + \hat{t} - 1$ of $\bar{S}_k$ one has that

$$p_k^j = p_{\tilde{k},r}^{j-\hat{t}+r} \leq \left\lceil\frac{k-(\hat{t}-r)/\bar{t}+1}{2}\right\rceil \text{ and } s_k^j = s_{\tilde{k},r}^{j-\hat{t}+r} \leq k - (\hat{t}-r)/\bar{t}\,. \tag{B.136}$$

For each $m = 1, \bar{t}+1, 2\,\bar{t}+1, 3\,\bar{t}+1, \ldots, \hat{t}-r+1$ follows from the Inequalities (B.135) and (B.136) that one processor is available during the computational cycles $j = \bar{t}\,l_{\tilde{k},r} + m, \ldots, (\bar{t}+1)l_{\tilde{k},r} + \hat{t} - 1$. This processor can be used in the following way to increase the number of physical steps that are reversed and hence to create the desired parallel reversal schedule S_k. The free processor performs the reverse step $\bar{F}_{l_{\tilde{k},r}+\lfloor m/\bar{t}\rfloor}$ in the computational cycles $\bar{t}\,l_{\tilde{k},r} + \lfloor m/\bar{t}\rfloor\bar{t} + 1, \ldots, \bar{t}\,l_{\tilde{k},r} + (\lfloor m/\bar{t}\rfloor + 1)\bar{t}$. Then the recording step $\hat{F}_{l_{\tilde{k},r}+\lfloor m/\bar{t}\rfloor}$ is evaluated using the free processor during the computational cycles $\bar{t}\,l_{\tilde{k},r} + (\lfloor m/\bar{t}\rfloor + 1)\bar{t} + 1, \ldots, \bar{t}\,l_{\tilde{k},r} + (\lfloor m/\bar{t}\rfloor + 1)\bar{t} + \hat{t}$ and the forward sweep from the initial state 0 to state $l_{\tilde{k},r} + \lfloor m/\bar{t}\rfloor$ during the computational cycles $\bar{t}\,l_{\tilde{k},r} + (\lfloor m/\bar{t}\rfloor + 1)\bar{t} + \hat{t} + 1, \ldots, (\bar{t}+1)l_{\tilde{k},r} + \lfloor m/\bar{t}\rfloor(\bar{t}+1) + \bar{t} - 1$. This extension is possible $(\hat{t}-r)/\bar{t}$ times. Obviously, the checkpoint writing that copies the initial state 0 has to be performed in the computational cycle $(\bar{t}+1)l_k + \hat{t} - 1$ instead of $(\bar{t}+1)l_{\tilde{k},r} + r - 1$. The total number of physical steps that can be reversed with S_k is given by $l_{\tilde{k},r} + (\hat{t}-r)/\bar{t} = l_k$ because of Equality (4.30). The number of computational cycles in S_k equals properly $(\bar{t}+1)l_{\tilde{k},r} + \hat{t} - 1 + (\bar{t}+1)(\hat{t}-r)/\bar{t} = (\bar{t}+1)l_k + \hat{t} - 1$. Furthermore, it follows from Equations (B.135) and (B.136) as well as from

$$\left\lceil\frac{k-(\hat{t}-r)/\bar{t}+1}{2}\right\rceil + \frac{\hat{t}-r}{\bar{t}} \leq \left\lceil\frac{k+1}{2}\right\rceil + \left\lceil\frac{1}{2}\left\lfloor\frac{\hat{t}-1}{\bar{t}}\right\rfloor\right\rceil$$

that in each computational cycle of S_k the number of resources applied does not exceed k. Furthermore, it is shown that in each computational cycle the number of processors used is not greater than p_k^b, which completes the proof. ∎

Bibliography

[AHU74] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[BBCG96] M. Berz, C. Bischof, G. Corliss, and A. Griewank (eds.), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM Philadelphia, 1996.

[BC63] R. Bellmann and K.L. Cooke, *Differential-difference equations*, Academic Press, 1963.

[Ben73] C. Bennett, *Logical reversibility of computation*, IBM J. Research and Development **17** (1973), 525 – 532.

[Ben95] J. Benary, *DAP – Dresdener Adjungierten Parallelisierungsprojekt*, Preprint IOKOMO-05-1995, Techn. Univ. Dresden, 1995.

[Ben96] J. Benary, *Parallelism in the reverse mode*, Computational Differentiation: Techniques, Applications, and Tools (M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds.), SIAM Philadelphia, 1996, pp. 137 – 147.

[Cha98] I. Charpentier, *Checkpoints schemes for adjoint codes: Application to the meteorological model Meso-NH*, Tech. report, Projet IDOPT, LMC-IMAG, 1998.

[Eul68] L. Euler, *Institutiones Calculi Integralis*, primum ed., vol. XI, Opera Omnia, 1768.

[Evt91] Y.G. Evtushenko, *Automatic differentiation viewed from optimal control*, Automatic Differentiation of Algorithms: Theory, Implementation, and Application (A. Griewank and G. Corliss, eds.), SIAM Philadelphia, 1991, pp. 25 – 30.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM parallel virtual machine, a user's guide and tutorial for networked parallel computing*, MIT Press, Cambridge, Mass., 1994.

[GC96] V. Goldman and G. Cats, *Automatic adjoint modelling within a program generation framework: A case study for a weather forecasting grid-point model*, Computational Differentiation: Techniques,

Applications, and Tools (M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds.), SIAM Philadelphia, 1996, pp. 185 – 194.

[Gea71] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, 1971.

[GJU96] A. Griewank, D. Juedes, and J. Utke, *ADOL-C, A package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Software **22** (1996), 131 – 167.

[GPRS96] J. Grimm, L. Pottier, and N. Rostaing-Schmidt, *Optimal time and minimum space-time product for reversing a certain class of programs*, Computational Differentiation: Techniques, Applications, and Tools (M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds.), SIAM Philadelphia, 1996, pp. 95 – 106.

[Gri89] A. Griewank, *On automatic differentiation*, Mathematical Programming: Recent Developments and Applications (Amsterdam) (M. Iri and K. Tanabe, eds.), Kluwer Academic Publishers, 1989, pp. 83 – 108.

[Gri92] A. Griewank, *Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation*, Optimization Methods and Software **1** (1992), 35 – 54.

[Gri00] A. Griewank, *Evaluating derivatives: Principles and techniques of algorithmic differentiation*, Frontiers in Appl. Math., no. 19, SIAM, Philadelphia, 2000.

[GW00] A. Griewank and A. Walther, *Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation*, ACM Trans. Math. Software **26** (2000), 19 – 45.

[HNW96] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems*, 2nd revised ed., Computational Mechanics, No. 14, Springer-Verlag, Berlin, 1996.

[HP98] P. Hilton and J. Petersen, *A fresh look at old favourites: The Fibonacci and Lucas sequences revisited*, Australian Mathematical Society Gazette **25** (1998), 146 – 160.

[Knu73] D.E. Knuth, *The Art of Computer Programming*, Computer Science and Information Processing, vol. I, Addison-Wesley, Massachusetts, 1973.

[Kub98] K. Kubota, *A fortran77 preprocessor for reverse mode automatic differentiation with recursive checkpointing*, Optimization Methods and Software **10** (1998), 319 – 336.

[KW98] W. Klein and A. Walther, *Application of techniques of computational differentiation to a cooling system*, Preprint IOKOMO-05-1998, Techn. Univ. Dresden, 1998, To appear in Optimization Methods and Software.

[Lin76] S. Linnainmaa, *Taylor expansion of the accumulated rounding error*, BIT (Nordisk Tidskrift for Informationsbehandling) **16** (1976), 146 – 160.

[MO] A. Mauer-Oats, *Checkpointing for ADOL-C:* `checkpoint-1.0.2`, http://www.math.uiuc.edu/~mauer/checkpoint/index.html.

[SB90] J. Stoer and R. Bulirsch, *Numerische Mathematik 2*, Springer Verlag, Berlin, 1990.

[Tal91] O. Talagrand, *The use of adjoint equations in numerical modeling of the atmospheric circulation*, Automatic Differentiation of Algorithms: Theory, Implementations, and Application (A. Griewank and G. Corliss, eds.), SIAM Philadelphia, 1991, pp. 169 – 180.

[vdS93] J. van der Snepscheut, *What Computing is all about*, Texts and Monographs in Computer Science, Suppl. 2, Springer Verlag, Berlin, 1993.

[VO85] Y.M. Volin and G.M. Ostrovskii, *Automatic computation of derivatives with the use of the multilevel differentiation technique*, Computers and Mathematics with Applications **11** (1985), 1099 – 1114.

[WNZD95] Z. Wang, I.M. Navon, X. Zou, and F.X. Le Dimet, *A truncated Newton optimization algorithm in meteorology applications with analytic Hessian/vector products*, Comput. Optim. Appl. **4** (1995), 241 – 262.

Versicherung

Hiermit versichere ich, daß ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

I affirm that I have written this dissertation without any inadmissible help from any third person and without recourse to any other aids; all sources are clearly referenced. The dissertation has never been submitted in this or similar form before, neither in Germany nor in any foreign country.

Die vorgelegte Dissertation habe ich am Institut für Wissenschaftliches Rechnen der Technischen Universität Dresden unter der wissenschaftlichen Betreuung von Herrn Prof. Ph.D. Andreas Griewank angefertigt.

I have written this dissertation at the Institute of Scientific Computing, Technical University Dresden, under the scientific supervision of Prof. Ph.D. Andreas Griewank.

Dresden, den 14. Oktober 1999

HSSS AdminTools (c) 2001, last visited: Fri Mar 01 11:06:42 GMT+01:00 2002