MULTIPROCESSOR SCHEDULING WITH AVAILABILITY CONSTRAINTS

A Dissertation

by

LILIANA GENTIANA ALEX GRIGORIU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2010

Major Subject: Computer Science

MULTIPROCESSOR SCHEDULING WITH AVAILABILITY CONSTRAINTS

A Dissertation

by

LILIANA GENTIANA ALEX GRIGORIU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Donald Friesen
Committee Members,	Riccardo Bettati
	Jianer Chen
	Maurice Rojas
Head of Department,	Valerie Taylor

May 2010

Major Subject: Computer Science

ABSTRACT

Multiprocessor Scheduling with Availability Constraints. (May 2010)

Liliana Gentiana Alex Grigoriu, Diplom, Technical University of Berlin

Chair of Advisory Committee: Dr. Donald Friesen

We consider the problem of scheduling a given set of tasks on multiple processors with predefined periods of unavailability, with the aim of minimizing the maximum completion time. Since this problem is strongly NP-hard, polynomial approximation algorithms are being studied for its solution. Among these, the best known are LPT (largest processing time first) and Multifit with their variants.

We give a Multifit-based algorithm, FFDL Multifit, which has an optimal worst-case performance in the class of polynomial algorithms for same-speed processors with at most two downtimes on each machine, and for uniform processors with at most one downtime on each machine, assuming that $P \neq NP$. Our algorithm finishes within 3/2 the maximum between the end of the last downtime and the end of the optimal schedule. This bound is asymptotically tight in the class of polynomial algorithms assuming that $P \neq NP$. For same-speed processors with at most k downtimes on each machine our algorithm finishes within $(\frac{3}{2} + \frac{1}{2k})$ the end of the last downtime or the end of the optimal schedule. For problems where the optimal schedule ends after the last downtime, and when the downtimes represent fixed jobs, the maximum completion time of FFDL Multifit is within $\frac{3}{2}$ or $(\frac{3}{2} + \frac{1}{2k})$ of the optimal maximum completion time.

We also give an LPT-based algorithm, LPTX, which matches the performance of FFDL Multifit for same-speed processors with at most one downtime on each machine, and is thus optimal in the class of polynomial algorithms for this case. LPTX differs from LPT in that it uses a specific order of processors to assign tasks if two processors become available at the same time.

For a similar problem, when there is at most one downtime on each machine and no more than half of the machines are shut down at the same time, we show that a bound of 2 obtained in a previous work for LPT is asymptotically tight in the class of polynomial algorithms assuming that $P \neq NP$.

ACKNOWLEDGMENTS

Donald Friesen told me of many methods which can be used in this research area and discussed this research with me whenever I wished to talk about it. The reviewers of SIAM Journal of Discrete Optimization and Riccardo Bettati sent comments and suggestions which were incorporated in the first two chapters of this thesis.

I am grateful to professors Lively and Simmons for allowing me to use a desk in their office HRBB 333 for about eight months. Support for this research also came from my family and the Department of Computer Science at Texas A&M.

The open atmosphere at Texas A&M, where the professors are willing to help when approached with any technical question, resulted in a good studying environment.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Nonpreemptive scheduling of a set of tasks on multiple resources is a widely encountered problem. Applications range from assigning waiting airplanes to departure lanes, or assigning terminals to airplanes that need to be loaded, to scheduling tasks on computing units or packets waiting in a buffer to links of a multilink connection. The jobs are usually assumed to be given as an integer number of time units representing computing units on the slowest processor or other suitable units.

The multiprocessor scheduling problem, whether it is possible to nonpreemptively schedule a set of independent tasks on m processors to meet a given deadline (with m considered to be an input parameter) is strongly NP-hard [3], and so are most related problems. As a consequence, the study of this area has been mainly concentrating on approximation algorithms: the *largest processing time first* (LPT) algorithm was first proposed [5] and shown to have a makespan within 4/3 the optimal makespan, and later the MULTIFIT algorithm was considered [8], and was shown to have a better performance of 13/11 in [18].

Due to maintenance or failures, machines might exhibit periods of unavailability. A recent result on this subject was obtained by Sadfi *et al.* for nonpreemptive scheduling on one processor [12], where they give an approximation algorithm to minimize the total completion time with an error bound of 3/17.

A review of deterministic scheduling in 1997 is given by C.Y. Lee, L. Lei, and M. Pinedo in [9], while a review of scheduling with availability constraints is given by Sanlaville and Schmidt in [13].

This dissertation follows the style of *SIAM Journal of Discrete Optimization.*

We focus on the static variant of the problem, when downtimes are known in advance. A dynamic variant can also be conceived, when downtimes can occur unexpectedly. In the case of nonpreemptive scheduling, the interruption of a task at the beginning of a downtime would lead to its reexecution from scratch, either by adding the task back to the set of tasks that need to be assigned to machines, or by waiting until the machine to which it was assigned starts processing again and processing it there. In the first case we have a variant of online scheduling, since tasks can be added to the set of tasks to be assigned at runtime.

While our focus is on offline scheduling, results have been obtained for online scheduling as well. For two identical-speed machines with availability constraints, online scheduling has been studied by Tan and He in [16]. They give an optimal algorithm to minimize the maximum completion time for a fixed set of jobs, for the situation when each machine shuts down only once, and the unavailability periods do not overlap.

A special case for scheduling on multiple processors in the presence of machine shutdowns is the case when all downtimes are at the beginning of the schedule, that is when the processors start processing at different times. C.Y. Lee [10] and Chang and Hwang [1] give worst-case analyzes of the multiprocessor scheduling problem for scheduling parallel machines that do not start simultaneously, when using LPT and MULTIFIT respectively.

When downtimes are not necessarily at the beginning of the schedule, resumable and non-resumable scheduling can be considered.

In resumable scheduling tasks can be interrupted by a downtime, and then resume after the end of that downtime. Resumable scheduling is different from preemptive scheduling because tasks can only be interrupted by machine shutdowns. C.Y. Lee showed in [11] for this case that the makespan of LPT is at worst $\frac{m+1}{2}$ times as long as the optimal makespan when one machine never shuts down and all others shut down at most once.

Given that all downtimes could be infinite, the strong NP-hardness of multiprocessor scheduling results in the NP-hardness of the problem of finding an approximation algorithm that ends within a multiple of the time needed by the optimal schedule, unless assumptions about the downtimes are made. This can be shown by attaching to any multiprocessor scheduling problem example infinite downtimes that start on each processor at the deadline of the considered example. This reduces the multiprocessor scheduling problem to the problem of finding an approximation algorithm that ends within a multiple of the time needed by the optimal schedule for the multiprocessor scheduling with machine shutdowns problem. Prohibiting infinite downtimes does not allow for better results, as the lengths of the downtimes can be chosen to be longer than any multiple of the optimal schedule.

For the case when there is at most one downtime on each machine, the authors in [6] make the assumption that no more than half the machines are unavailable at any time. They show that for this situation the LPT algorithm ends within twice the time needed by the optimal schedule. In [7], the result is generalized to the case when an arbitrary number of machines, $\lambda \in 1, .., m-1$, can be unavailable at the same time. In that case the makespan generated by the LPT schedule is not worse than the tight worst-case bound of $1 + \frac{1}{2}[m/(m-\lambda)]$ times the optimal makespan.

In [14] Scharbrodt *et al.* give a polynomial-time approximation scheme for the problem of scheduling with "fixed" jobs, that is jobs that have to execute at certain predefined times. The approximation scheme is for minimizing the makespan of the schedule for all the jobs, it does not consider the number of processors as a part of the input, and there can be more than one fixed job on one machine.

In the case of same-speed processors, we first consider the problem of nonpreemptive (and nonresumable) scheduling of a set of independent tasks on multiple machines, each of which may become unavailable for an predefined period of time at most once. For this case, we give a polynomial algorithm, LPTX, (LPT with a specific ordering of processors to assign tasks when two processors become available at the same time), the schedule of which finishes within the latest among 3/2 the optimal maximum completion time or 3/2 the time until the end of the last downtime, if there is at most one downtime on each machine. This implies that, when LPTX finishes after 3/2 the end of the last downtime, or when the downtimes represent fixed jobs, it also finishes within 3/2 the end of the optimal schedule. Also, if there is another way of determining that the optimal schedule ends after the end of the last downtime, such as when the sum of all task and downtime lengths is greater then the end of the last downtime multiplied by m, then LPTX finishes within 3/2 the optimal schedule length. Recall that m is the number of processors in the problem instance. In cases where it can not be determined in polynomial time whether the optimal schedule ends after the last downtime, or if the optimal schedule ends before the end of the last downtime, our result has no immediate implication on whether LPTX finishes within 3/2 the optimal schedule length or not.

We also consider the case when there can be multiple downtimes on one machine. We give a Multifit-based algorithm which finishes finishes within $\frac{3}{2} + \frac{1}{2k}$ the optimal schedule or the end of the last downtime when there are at most k downtimes on each machine. The classic Multifit algorithm assigns upper and lower bounds for the schedule length, then uses a binary search while assigning schedule deadlines between the initially chosen upper and lower bound. It uses the First Fit Decreasing (FFD) algorithm to assign tasks to the time slots resulting from an deadline assigned by Multifit and he start of the schedule, and if a feasible schedule is found the deadline is decreased, and otherwise it is increased, until a desired accuracy is achieved. Our algorithm orders the time slots formed by the times between the start of the schedule and the downtimes, by the times between the downtimes, and by times that start at the end of a downtime and end at the Multifit assigned deadline in increasing order before assigning tasks to time slots. We also show that the bound can be improved when there are at most 2 downtimes on each machine. For this case the bound is 3/2, and this implies that our algorithm is optimal in the class of polynomial algorithms when there are at most two downtimes on each machine assuming $P \neq NP$.

Unlike in [6], we do not have any restriction on the times when the machines shut down. The problem in [14] is similar to our problem in that the machine downtimes are equivalent to the fixed jobs. The difference is that the maximum completion time of the optimal schedule with fixed jobs cannot be less than the maximum completion time of the last fixed job, which corresponds to the end of the last downtime in our setting.

Scheduling on uniform processors has also been studied extensively in the past. In [2], the authors show that for nonpreemptive scheduling a variant of Multifit finishes within 1.4 the optimal schedule length. In [17], the performance of LPT for scheduling on uniform processors with nonsimultaneous machine available times is studied, and it is shown that LPT finishes within 5/3 the optimal schedule length, and that the bound is better when there are only 2 machines or when the speed ratio is small. The paper also presents a polynomial algorithm that finishes within 6/5 the optimal schedule length if there are only 2 processors in the system. In this work, we consider the more general case when machines may have a downtime which does not necessarily start at the beginning of the scheduling period. We give a polynomial algorithm that finishes within $\frac{3}{2}$ the optimal schedule length or the end of the last downtime independent of the speed ratios are or the number of processors in the system. Our algorithm is shown to be optimal in the class of polynomial algorithms for scheduling on uniform processors with at most one shutdown on each machine assuming $P \neq NP$.

According to the notation used in [15] the problem with same-speed processors can be classified as $(P, NC_{win}||C_{max})$: scheduling on multiple machines the number of which is not fixed and which are not continuously available, with given tasks that cannot be interrupted and are available at time 0, while minimizing the maximum completion time. To indicate a maximum number of unavailability periods on each machine, an integer can be added after NC_{win}. For example, when there are at most k downtimes on each machine, we would have the notation $(P, NC_{win}k||C_{max})$. The uniform processor scheduling problem can be classified as $(Q, NC_{win}1||C_{max})$.

In addition to the upper bound results we show that in the cases we consider the bound of 3/2 is asymptotically tight in the class of polynomial algorithms assuming that $P \neq NP$ (our proof bases on the proof presented in [15]). We also prove that the bound of 2 for scheduling on same-speed processors with at most one shutdown on each machine when at most half of the machines are shut down at any time, which was obtained in [6], is asymptotically tight in the class of polynomial algorithms assuming that $P \neq NP$.

Chapter II considers scheduling on same-speed processors with at most one shutdown time on each machine, and contains the tightness results. In Chapter III we consider scheduling on uniform processors with at most one downtime or fixed job on each machine. In Chapter IV we analyze the performance of an adapted FFDL Multifit, the algorithm we present in Chapter III, for scheduling on same-speed processors in the case when there are multiple downtimes on each machine. Chapter V contains concluding remarks and an enumeration of the most important methods used in our proofs.

CHAPTER II

SCHEDULING ON SAME-SPEED PROCESSORS WITH AT MOST ONE DOWNTIME ON EACH MACHINE

In this chapter we introduce the LPTX scheduling algorithm, and show that it finishes within $\frac{3}{2}$ the end of the optimal schedule or the end one the last downtime. We also show that the bound is tight in the class of polynomial algorithms assuming that $P \neq NP$. The next subsection contains the upper bound result while subsection B contains tightness results. Subsection C contains some conclusive remarks.

A. Upper bound for LPTX Schedule Length

In this subsection we introduce the LPTX scheduling algorithm, and we show that its schedule finishes within 3/2 of the optimal schedule's end or of the end of the last downtime.

The LPTX scheduling algorithm starts with ordering the processors, such that the LPT algorithm breaks ties among processors that are available at the same time in a predefined way. Given a set of processors P with downtimes starting at time δ_p and ending at time γ_p for each processor $p \in P$, and a set of tasks T, the LPTX algorithm is:

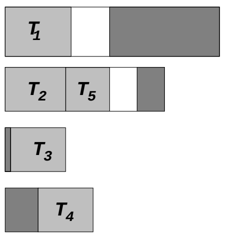

Fig. 1. Order of processors in which LPTX assigns tasks if two processors are available at the same time; darker shaded areas represent the downtimes.

```
LPTX($P, T$)
Initialize $P_1$ and $P_2$ as empty lists
for $p \in P$
    if ($\delta_p > 0$)
        append $p$ to $P_1$;
    else
        append $p$ to $P_2$.
end for
Sort $P_1$ by $\delta_p$ in increasing order
Sort $P_2$ by $\gamma_p$ in increasing order
$P' = P_1 o P_2$
LPT($P', T$)
```

When a processor has no downtime, we assume $\delta_p = \gamma_p = 0$. After sorting the ordered sets P_1 and P_2 in increasing order of δ_p and γ_p respectively, the algorithm executes a concatenation of the two lists and saves it in a list P'. Then the LPT-algorithm is used to assign tasks to processors. Recall that LPT sorts the tasks in decreasing order of their required processing time and assigns them to the first processor on which they can be processed at the earliest time. Ties are broken by the order in which the processors are ordered in the input list. For LPTX, this order is also represented in Figure 1.

This algorithm is polynomial if all comparisons necessary to sort the tasks and all operations necessary to compute the next available processor can be done in polynomial time. This condition is given when all arguments are integers or rational numbers.

The task assignment strategy was mainly chosen to facilitate the proof. It has some similarity to best fit bin packing for the assignment of the first task.

We call pretime the available time of a processor before its downtime starts. The length of a pretime of a processor p will be denoted with pre_p, and the start and end of its downtime with δ_p and γ_p respectively. Figure 2 shows a possible LPTX schedule of a processor p.

We denote with *lpt* the end of the LPTX-schedule and with *opt* the end of the optimal schedule.

Most of the remainder of this subsection is devoted to proving the following theorem:

Theorem A.1 (Bound for LPTX-schedules)

The maximum completion time of an LPTX-schedule is less or equal to 3/2 the maximum completion time of the optimal schedule or 3/2 the maximum end of a downtime,

$$lpt \leq \frac{3}{2} max(opt, max_{p \in P}(\gamma_p)).$$

First we will define a minimal counterexample, which is shown to exist whenever there is a counterexample. Then we prove several properties of a minimal counterexample, at last resulting in the fact that such a counterexample does not exist. Several theorems and lemmas contribute to this proof.

In subsection B we will show this bound to be asymptotically tight within the class of polynomial algorithms assuming that $P \neq NP$.

A *problem instance* (P, T) is given by a set P of processors with their downtimes and a set T of tasks with their durations. We denote the last task scheduled by the LPTX-schedule with $\overline{X}$. In a minimal counterexample this will be the task that breaks the 3/2 bound.

In the following we shall assume that the 3/2 bound is broken and derive a contradiction.

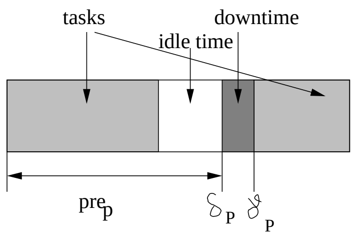

Fig. 2. LPTX-schedule of a processor p

Definition A.2 (Order relation on problem instances)

Given two problem instances C1=(P1,T1), C2=(P2,T2) where T1 and T2 are sets of tasks with their execution times and P1 and P2 sets of processors with downtimes we say that $C1 < C2$ if any of the following holds:

a) $|T1| < |T2|$

b) $|T1| = |T2|$ and $|P1| < |P2|$

c) $|T1| = |T2|$, $|P1| = |P2|$, and the number of processors with pretimes in C1 is less than the number of processors with pretimes in C2.

Here $|S|$ represents the number of elements in a set S.

Definition A.3 (Minimal Counterexample)

A minimal counterexample is a pair C= (P,T) where P is a set of processors with downtimes, and T is a set of tasks with their execution times such that the LPTX schedule exceeds $\frac{3}{2}opt$ and $\frac{3}{2}\underset{p\in P}{max}(\gamma_p)$, such that C is minimal with regard to the order relation defined in Definition A.2. Recall that γ_p denotes the end of the downtime of processor $p \in P$.

If there is a counterexample then there also is a minimal counterexample, according to the following argument. Suppose we have a set S of counterexamples. Then there must be a subset S1 of counterexamples that have a minimum number of processors. Among those there must be a subset with a minimum number of tasks.

There must be a subset of counterexamples of this last set that have a minimum number of processors with pretimes.

From now on we will assume that the counterexample we are considering is minimal.

Recall that $\overline{X}$ is the task in the LPTX-schedule that breaks the 3/2 bound. All tasks that the LPTX schedule would schedule after $\overline{X}$ are irrelevant to the fact that the LPTX schedule breaks the bound, thus a minimal counterexample only contains tasks that have a length that is greater or equal to that of the task that breaks the bound and that task is the last task scheduled by the LPTX algorithm.

Notation A.4 (Measure of time)

In the following we normalize the length of every time interval to the length of the last task $\overline{X}$, that is, a number will represent that same number times the task length $\overline{X}$ in the measuring of time.

We continue by showing some more properties.

Lemma A.5 ($\geq$ 1 tasks in LPTX pretimes)

In a minimal counterexample the LPTX-schedule has at least one task in the pretime of each processor that has a pretime.

Proof: If the pretime of a processor p is empty in the LPTX-schedule, then the last task $\overline{X}$ did not fit in that pretime. But $\overline{X}$ is the least task, and thus the optimal schedule could also fit nothing in that pretime. Then we can build a lesser counterexample by maintaining the same processors and tasks with the difference that the pretime of p is replaced by downtime. Both the optimal schedule and the LPTX schedule will remain the same, and the new counterexample has fewer pretimes. $\triangle$

Lemma A.6 (2 tasks in optimal pretimes)

The optimal schedule of a processor with a pretime has at least two tasks in the pretime.

Proof: Suppose this is not the case and that we have a minimal counterexample (P,T) where a processor $p \in P$ has only one task X in its pretime in the optimal schedule.

The LPTX-schedule must have at least one task on p (by Lemma A.5). Let X' be the first task in the LPTX-schedule of p. If $X' \geq X$ then we can get a lesser counterexample by removing X' and any other tasks scheduled in the pretime of p by LPTX, and by filling the pretime of p with downtime: in the optimal schedule X can be put where X' was before resulting in a schedule for the new set of tasks and processors that is at least as good as in the initial example, and the LPTX-schedule does not change. If X' is the same as the task X then the problem of finding a place for X in the optimal schedule disappears as X' is removed from the task set.

Thus $X' < X$. Let q be the processor on which LPTX has scheduled X. Then $pre_q \leq pre_p$, else X would have been scheduled by LPTX on p. Removing X and any other tasks scheduled by LPTX in the pretime of q from the task set and filling the pretime of q with downtime we get a lesser counterexample: there are less tasks, *opt* will stay the same in the worst case, since we can move anything that was scheduled in the pretime of q to the pretime of p, and the LPTX schedule will stay the same. Note that $\overline{X}$ can not be in the pretime of any machine, since it is supposed to finish after the end of all pretimes, and thus its position is not affected by the above changes.

$\triangle$

Lemma A.7 (Corollary)

The pretimes end at or after time 2 for all processors with pretimes.

Lemma A.8

There are processors with (nonzero) pretimes.

Proof: In [10] C.Y. Lee has shown that the LPT schedule for a multiprocessor scheduling problem with nonsimultaneous processing start times has a makespan bounded by $\frac{3}{2} - \frac{1}{2m}$, where m is the number of processors. Thus, since our algorithm uses LPT after ordering the processors, any counterexample must have at least one processor with a (nonzero) pretime. △

Lemma A.9 (Tasks different from $\overline{X}$ after each downtime in LPTX)

In the LPTX-schedule of a minimal counterexample there are tasks different from the last scheduled task $\overline{X}$ after each downtime. That is, $\overline{X}$ is scheduled on top of a task that is scheduled after a downtime.

Proof: Let q be the processor the downtime of which ends last.By Lemma A.7 all downtimes end after time 2 if there is a pretime. By Lemma A.8 there are pretimes. So $\gamma_q > 2$, and $\gamma_q + 1 < \frac{3}{2}\gamma_q \leq \frac{3}{2}(max(\underset{p \in P}{max}(\gamma_p), opt))$. Thus $\overline{X}$ starts executing after γ_q, else LPTX would end at or before time $\frac{3}{2}(max(\underset{p \in P}{max}(\gamma_p), opt))$, and we would not have a counterexample. Thus there must be another task between $\overline{X}$ and the end of any downtime. △

Lemma A.10 (The start of $\overline{X}$)

The start time L of the last task $\overline{X}$ in the LPTX schedule is greater than $\frac{3}{2}opt - 1$. Also, the LPTX-schedule must be busy until that time on all processors (except for the idle time in the pretimes). Also, $L > 2$, $L > opt$, and

$$L - opt > \frac{1}{2}opt - 1$$

Proof: The LPTX-schedule ends when task $\overline{X}$ ends, thus $lpt = L + 1$ Since $lpt > \frac{3}{2}opt$ we have $L > \frac{3}{2}opt - 1$. If the LPTX-schedule on a processor would end before time L, the LPTX-algorithm would have scheduled $\overline{X}$ there and not at time L. Since there are processors with pretimes, Lemma A.6 we have *opt* ≥ 2, thus $L > \frac{3}{2}opt - 1 \geq 2$. We also have $L > \frac{3}{2}opt - 1 \geq \frac{3}{2}opt - \frac{1}{2}opt = opt$, proving $L > opt$. Last, since the $\frac{3}{2}$ bound is broken $L = lpt - 1 > \frac{3}{2}opt - 1$, implying $L - opt > \frac{3}{2}opt - 1 - opt$, and $L - opt > \frac{1}{2}opt - 1$. $\triangle$

Lemma A.11 (opt$\geq$ 3)

The length of the optimal schedule is greater or equal to 3.

Proof: Suppose this is not true and let (P, T) be a counterexample where the optimal schedule OPT has a length *opt* < 3. We consider the LPTX-schedule of $(P, T \backslash \{\overline{X}\})$, denoted $LPTX - \overline{X}$ for legibility, and show that this schedule has at least as many tasks as the optimal schedule of (P, T), a contradiction. We denote with $T_{LPTX-\overline{X}}$ the number of tasks in the LPTX-schedule excluding $\overline{X}$, and with T_{OPT} the number of tasks in the optimal schedule. To show is

$$T_{LPTX-\overline{X}} \geq T_{OPT}.$$

For a particular processor p the inequality becomes $T_{LPTX-\overline{X}}(p) \geq T_{OPT}(p)$, and for a set P^* of processors we write the inequality as $T_{LPTX-\overline{X}}(P^*) \geq T_{OPT}(P^*)$. The inequality holds for each processor that has a pretime, since $T_{LPTX-\overline{X}}(p) \geq 2$ as shown above, and $T_{OPT}(p) \leq 2$, since we assumed *opt* < 3.

For processors with no pretime, the downtime of which ends after time 1, the inequality holds again, since the optimal schedule can't put more than one task on them, and $LPTX - \overline{X}$ must have at least one task on them by Lemma A.9.

Let P^* be the set of the remaining processors (with no pretime, and with $\gamma_p < 1$). If $LPTX$ has two tasks on such a processor p the inequality holds for p since OPT can't put more than two tasks on it. If p has only one task Y in its $LPTX$-schedule, then $Y > L - \gamma_p > L - 1 > opt - 1$, since $\gamma_p < 1$ and $L > opt$. Thus OPT must schedule Y alone on a processor q ($Y > opt - 1$), and this processor can't have a pretime (on processors with pretimes OPT has two tasks), and $\gamma_q < 1$, so $q \in P^*$. The inequality to prove follows for P^*, as there are at least as many processors with only one task on them in the optimal schedule as there are in the LPTX-schedule.$\triangle$

Lemma A.12 (Idle times in LPTX schedules)

The idle time in the LPTX-schedule of the pretime of any processor is shorter than 1.

Proof: Suppose this is not the case and we have a processor p on which this situation is encountered. We know by Lemma A.9 that the last task $\overline{X}$ starts after the end of all pretimes. Thus at the time when $\overline{X}$ is scheduled the LPTX-algorithm would first try to fit it in a pretime, and succeed in doing that in the idle time of p. Thus $\overline{X}$ can't have been scheduled after the end of the last downtime, which contradicts Lemma A.9. $\triangle$

Next we derive a lemma concerning the difference between the end of the optimal schedule and the time L defined in Lemma A.10.

Lemma A.13 ($opt < 4$)

The length of the optimal schedule is less than 4.

Proof: Suppose there is a minimal counterexample with $opt \geq 4$. Then by Lemma A.10 we have $L - opt > \frac{1}{2}opt - 1 \geq 1$, and so the LPTX-schedule ends on each processor after a time interval of length 1 after the end of the optimal schedule. This additional busy time, however, must be compensated by busy time in the optimal schedule that occurs at a time when the LPTX-schedule is idle, before the downtimes. This implies that the LPTX-schedule contains in total before its downtimes an idle time of length at least $|P|$, which averages in an idle time of length 1 per processor, contradicting Lemma A.12, which states that all idle times are less than 1. $\triangle$

To better describe schedules on processors we will use the following notation for each processor schedule: [will denote start of the schedule, time 0, | will represent the downtime, and $[A_1A_2\dots A_n|B_1B_2\dots B_m$ will denote a schedule that has the tasks $A_1, A_2, \dots A_n$ in the pretime in the given order and the tasks $B_1, B_2, \dots B_m$ in the given order after the pretime.

We call Y-tasks all tasks the length of which is in the interval $[1.5, 2)$, and R-tasks all tasks of length 2 or longer.

Also, we denote with $busy_{ALG}(p)$ length of the total processing time of processor p in an ALG-schedule, and with $LPTX-\overline{X}$ the LPTX-schedule of $(P, T\backslash\{\overline{X}\})$, where (P, T) is the considered problem instance. For a processor p, $OPT(p)$ and $LPTX(p)$ denote its optimal schedule and respectively its LPTX-schedule.

Theorem A.14 (Constraint for LPTX-schedule)

If there is a minimal counterexample, then there is at least one processor p the LPTX-schedule of which is one of the following:

(a) $[Y_1|Y_2$ or

(b) $[|Y_1Y_2$ with $busy_{LPTX}(p) > 3.5 + \frac{3}{2}\gamma_p$ (this implies $Y_1 > 1.75 + \frac{3}{4}\gamma_p$),

where Y_1 and Y_2 are Y-tasks, that is $Y_1, Y_2 \in [1.5, 2)$.

The last Y-task in the LPTX-schedule is scheduled after time 1.75.

Proof: We use a weighing argument. Consider the following task categories: $X \in [1, 2)$ with $w(X) = 1$ and $R_1 \in [2, 3.5)$ with $w(R_1) = 2$. All longer tasks are called R_2 and have a weight of 3. Suppose (a) and (b) do not occur in the $LPTX$ schedule.

We consider the total weight of the tasks in the LPTX-schedule omitting $\overline{X}$, and compare it to the total weight of the tasks in the optimal schedule. We do this by considering each processor separately. For a given processor p we denote with

$w_p(ALG)$ the sum of the weights of the tasks scheduled by the algorithm ALG on the processor p.

We show that $w_p(LPTX - \overline{X}) \geq w_p(OPT)$ for each processor p which leads to a contradiction since we should have $w(LPTX) = w(OPT)$. Since $opt < 4$, we have $w_p(OPT) \leq 3$ for all processors. To consider is the case when $w_p(LPTX - \overline{X}) < 3$ on a processor p. In this case the LPTX-schedule of p has no R_2-tasks.

If p has a pretime, then we could have the situation $[X_1|X_2$ (X_1 and X_2 are X-tasks) in the $LPTX - \overline{X}$ schedule. Both tasks need to be there by Lemmas A.5 and A.9. The optimal schedule has two tasks in the pretime so it could have schedules $[XX|$ or $[XX|X$. $[R_1X|$ is impossible since then the LPTX schedule would also have another task in the pretime after X_1, as $\overline{X}$ would fit. If $OPT = [XX|$ we have $w_p(OPT) = 2$ and then $w_p(LPTX - \overline{X}) \geq w_p(OPT)$. So $OPT(p) = [X_3X_4|X_5$, for some X-tasks X_3, X_4 , and X_5. We denote with $end_{LPTX}(X)$ the time when task X ends in the LPTX schedule, and with $end_{OPT}(X)$ the time when task X ends in the optimal schedule. We have $end_{LPTX}(X_2) \geq L$ and $end_{OPT}(X_5) \leq opt$, and thus $end_{LPTX}(X_2) - end_{OPT}(X_5) \geq L - opt$, and $X_2 - X_5 \geq L - opt \geq 0.5$, by Lemma A.10 and Lemma A.11. Recall that in order for the weight of X_2 to be 1 it had to have length less than 2. Thus X_2 is a Y-task and we have case (a). (X_1 must be a Y-task as well since if it were shorter than X_2, then X_2 would not have been scheduled outside the pretimes by LPTX, because $pre_p \geq 2$ by Lemma A.6.) Also, X_2 is a Y-task scheduled after time $2 > 1.75$.

If a processor p with $w_p(LPTX - \overline{X}) < w_p(OPT)$ has no pretime, then $w_p(OPT) > w_p(LPTX - \overline{X}) \geq 1$, so $busy_{OPT}(p) \geq w_p(OPT) \geq 2$ and $L \geq opt \geq busy_{OPT}(p) + \gamma_p \geq 2 + \gamma_p$. Thus the busy time of $LPTX$ on processor p before $\overline{X}$ starts is greater than 2, and LPTX needs 2 X-tasks or an R_1-task to fill it. So $w_p(LPTX - \overline{X}) \geq 2$. In order for OPT to have a greater weight, we need $OPT(p) = [|XXX$ (or $OPT(p) =$

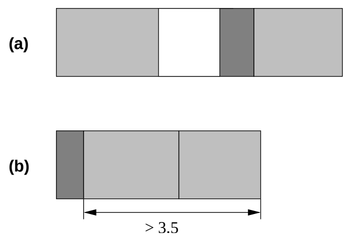

Fig. 3. Cases (a) and (b) from Theorem A.14

$[|R_1X$ or $[|R_2)$, and so $opt \geq 3 + \gamma_p$. Then $L > \frac{3}{2}opt - 1 \geq \frac{3(3+\gamma_p)}{2} - 1 = 3.5 + \frac{3}{2}\gamma_p$. Thus if $w_p(LPTX - \overline{X}) < 3$ we need $[|X_1X_2$ with both $X_1 < 2$ and $X_2 < 2$ (else the sum of their weights will be greater).

Also $X_1 + X_2 \geq L$, and thus $X_1 \geq 1.75 + \frac{3}{4}\gamma_p$, since it was scheduled before X_2. Also since $X_1 < 2$, and $X_1 + X_2 > 3.5$ we need $X_2 > 1.5$, and thus both X_1 and X_2 are Y-tasks. We have case (b). △

The two cases one of which must occur according to the previous theorem are shown in Figure 3.

The previous Theorem shows that the last Y-task appears after 1.75 in the LPTX-schedule. Together with Lemma A.7, which states that pretimes are longer or equal to 2, we know that the LPTX schedule has at least one task longer than or equal to 1.5 in each pretime, which we state in the following Corollary.

Corollary A.15 (Long tasks in LPTX-pretimes)

In each LPTX-pretime there is a task the length of which is at least 1.5.

Definition A.16 (Compensating processor)

A processor that has less busy time in the LPTX-schedule, without considering the last task $\overline{X}$, than in the optimal schedule is called a compensating processor.

Lemma A.17 (Existence of compensating processors)

There is at least one compensating processor.

Proof: The total busy time of the LPTX-schedule without the last task $\overline{X}$ is less than the total busy time of the optimal schedule since this one contains task $\overline{X}$. Thus there must be a processor on which OPT has more busy time than LPTX without task $\overline{X}$.

Lemma A.18 (Structure of a compensating processor)

Let p be a compensating processor. Then p has a pretime and:

(a) the optimal schedule on p is of the following form: $[X_1X_2|X_3$, where X_1, X_2, and X_3 are arbitrary tasks.

(b) the LPTX schedule on p is of the form $[Y|X_4X_5$, where Y is a Y-task, and $X_4, X_5 \in T$.

Proof: If p does not have a pretime, then $busy_{LPTX}(p) - busy_{OPT}(p) \geq L - opt > \frac{1}{2}opt - 1$ and p is not compensating. If p has a pretime, then the LPTX schedule on p has more busy time than the length of the pretime. Thus the optimal schedule on p must have a task X_3 after the downtime. We already know from Lemma A.6 that the optimal schedule has two tasks in the pretime and (a) follows. From Corollary A.15, we know that the LPTX schedule on p has a task ≥ 1.5 in the pretime. The busy time after the downtime in the LPTX schedule is $> X_3 + 0.5opt - 1 \geq 1.5$. If there were two tasks in the pretime of the LPTX-schedule, the busy time would be $\geq 1.5 + 1 + 1.5 = 4$ and p could not be compensating by Lemma A.13. Thus there is only one task, Y_1, during the pretime. If there were only one task, Z, after the downtime, then there are two cases depending on whether Z would fit in the pretime.

Case 1. If $Z \leq pre_p$, then $Y_1 \geq Z > X_3 + 0.5opt - 1 \geq 0.5opt$ and $Z + Y_1 > 2(0.5opt) = opt$ and p is not compensating. This argument also works if Y_1 is an R-task, since then Y_1 is still greater than Z.

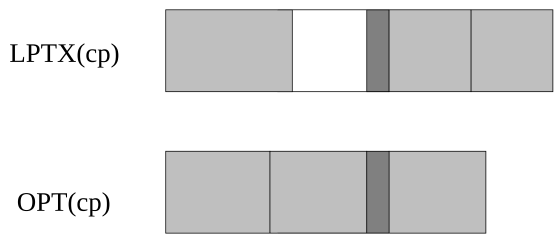

Fig. 4. Schedules of a compensating processor

Case 2. If $Z > pre_p$, then $Z > X_1 + X_2$ and so $X_3 > Y_1 \geq 1.5$. Then $Z + Y_1 > X_3 + L - opt + Y_1 \geq X_3 + 0.5opt - 1 + Y_1 \geq 2Y_1 - 1 + 0.5opt \geq 2 + 0.5opt > opt$ and again p is not compensating. Thus LPTX scheduled two tasks after the downtime of p.

Next we consider the size of Y_1. If $Y_1 \geq 2$, then $busy_{LPTX}(p) \geq 4$, and the optimal schedule can not be longer than that. Thus Y_1 cannot be an R-task. This completes the proof of (b). $\triangle$

Possible optimal and LPTX schedules of a compensating processor are represented in Figure 4. Next, we show that no minimal counterexample exists, completing the proof of Theorem A.1.

Proof: We show that a compensating processor can not coexist with situations (a) and (b) in Theorem A.14. Suppose they can coexist.

Let the LPTX-schedule of a compensating processor cp be $[Y_{cp}|X_4X_5$ where Y_{cp} is a Y-task. We can't have $[Y_{cp}|R_{cp}$ because of Lemma A.18. Also, let $[X_6X_7|X_8$ be $OPT(cp)$. We thus have

$$busy_{LPTX}(cp) > 2 + Y_{cp},$$

since the processor is compensating.

We consider the cases from Theorem A.14 (also see Figure 3):

(a) There is a processor p with $[Y_1|Y_2$ in the LPTX-schedule, and $[X_1X_2|X_3$ in the optimal schedule. $Y_2 - X_3 \geq L - opt > \frac{1}{2}opt - 1$. This is because Y_2 and X_3 start at the same time and the end of Y_2 occurs after L (or at time L) and the end of X_3 occurs before the end of the optimal schedule or at that time. The second inequality results from Lemma A.10. Then $Y_2 > X_3 + \frac{1}{2}opt - 1 \geq \frac{1}{2}opt$. From the order in which LPTX assigns tasks to processors we know that $Y_{cp} \geq Y_2$, and thus $Y_{cp} > \frac{1}{2}opt$. We have

$$busy_{OPT}(cp) > busy_{LPTX-\overline{X}}(cp) \geq Y_{cp} + 2 > 2Y_{cp} \geq opt,$$

a contradiction.

(b) $[|Y_1Y_2$ with $busy_{LPTX-\overline{X}}(p) > 3.5 + \frac{3}{2}\gamma_p$,
where Y_1 and Y_2 are both Y-tasks, that is $Y_1, Y_2 \in [1.5, 2)$. Recall that $OPT(cp) = [X_6X_7|X_8$, and thus $opt \geq \gamma_{cp} + 1$. Also since $opt < 4$, and cp is compensating, we can't have $busy_{LPTX-\overline{X}}(cp) \geq 4$, and thus $X_4 + X_5 < 4 - Y_{cp} \leq 2.5$, so $X_4 < 1.5 \leq Y_2$. The start time of Y_2 must be before the start time of X_4 due to the LPTX scheduling policy, and thus Y_2 starts before time γ_{cp}. Also Y_2 must end after time L. So

$$Y_2 > X_8 + L - opt > 1 + \frac{1}{2}opt - 1,$$

by Lemma A.10 and because $X_8 \geq 1$. So $Y_2 > \frac{1}{2}opt$. Also, $Y_2 \leq Y_{cp}$ by LPTX scheduling policy, and so $Y_{cp} > \frac{1}{2}opt$ implying a contradiction as in the previous case. $\triangle$

We give an example that shows that the bound above is asymptotically tight. The two schedules are represented in Figure 5. There are two processors, the downtime of the first processor starts at time 1, and ends at time $1+\epsilon$, the second processor has no downtime, and the tasks have lengths of $T_2 = \frac{1}{2}, T_3 = \frac{1}{2}, T_4 = \frac{1}{2}$, and $T_1 = \frac{1}{2}+\epsilon$ respectively. Then the quotient between the time needed by the LPTX-schedule and

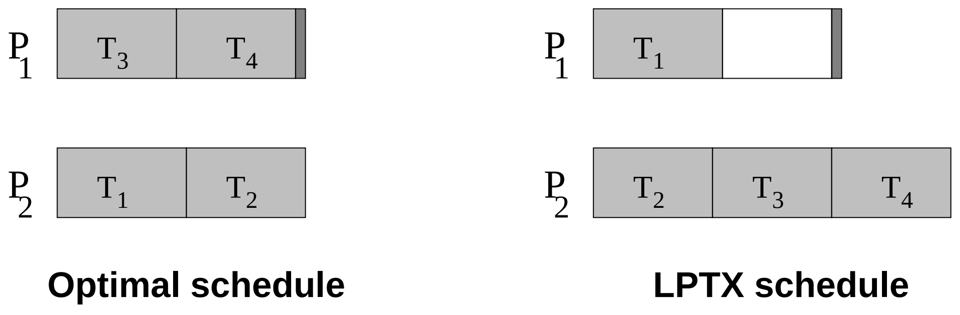

Fig. 5. Example showing that the 3/2 bound is asymptotically tight

that needed by the optimal schedule is $\frac{3}{2(1+\epsilon)}$. Since ϵ can be arbitrarily small, the bound proved in Theorem A.1 is asymptotically tight.

The bound in Theorem A.1 implies that whenever $lpt > \frac{3}{2}\gamma_p$ for all $p \in P$, we also have $lpt < \frac{3}{2}opt$. Also if there is some other way to conclude that $opt \geq max_{p \in P}(\gamma_p)$, such as when the sum of the task lengths and the downtime lengths divided by $|P|$ is greater or equal to $max_{p \in P}(\gamma_p)$, then it can also be determined that $lpt \leq \frac{3}{2}opt$ from this Theorem.

B. Asymptotically tight lower bounds for scheduling with machine shutdowns

In this subsection we show that the bound derived in subsection II is asymptotically tight within the class of polynomial algorithms (assuming that $P \neq NP$), and that the bound derived in [6] for the performance of LPT with respect to a related problem is also tight.

To this end, we first derive the NP-hardness of a problem we *called* 3-Partition with fixed bottom elements, that is a restatement of Numerical Matching with Target Sums, which has been shown to be NP-complete in [4]. Then we proceed with the proofs.

We next state Numerical Matching with Target Sums, as given in [4].

Definition B.1 (Numerical Matching with Target Sums (NMTS))

INSTANCE: Disjoint sets X and Y, each containing m elements, a size $s(a) \in Z^+$ for each element $a \in X \cup Y$, and a target vector $< B_1, B_2, \ldots B_m >$, with positive integer entries.

QUESTION: Can $X \cup Y$ be partitioned into m disjoint sets $A_1, A_2, \ldots A_m$, each containing exactly one element from each X and Y, such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B_i$?

Definition B.2 (3-Partition with fixed bottom elements (3PFB))

INSTANCE: A finite set A of $3n$ elements, and a subset Q of n distinguished elements of A, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$, such that $s(A)$ satisfies $B/4 < s(a) < B/2$, and such that $\sum_{a \in A} s(a) = nB$,

QUESTION: Can A be partitioned into n disjoint sets $S_1, S_2, \ldots, S_n$ such that for $1 \leq i \leq n$, $\sum_{A \in S_i} s(A) = B$, and each set S_i contains exactly one distinguished element?

The following lemma states that the 3-Partition with fixed bottom elements problem is NP-hard.

Lemma B.3

3-Partition with fixed bottom elements is NP-hard.

Proof: Follows directly from the NP-hardness of NMTS. △

Next we show the asymptotical tightness of the $\frac{3}{2}$-bound obtained in the previous section within the class of polynomial algorithms assuming that $P \neq NP$. In [14], the authors prove that the scheduling with fixed jobs problem when one machine can have more than one fixed job, can not be solved in polynomial time within $3/2 - \epsilon$ times the optimal makespan, and while doing that they do not use the assumption that machines can have more than one fixed job. Since fixed jobs are interchangeable with shutdown times in this context, their proof, which they say was suggested by Gerhard Woeginger, also results in the following theorem.

Theorem B.4 (Tightness of the $\frac{3}{2}$ bound)
If $P \neq NP$ then no polynomial algorithm can always produce a solution that ends before $k * opt$ for a constant $k < \frac{3}{2}$, and where *opt* is the time when the optimal schedule ends.

In the following we refer to the problem considered by [6] where the authors have studied how well LPT performs for scheduling tasks on machines that have predefined shutdown times, assuming that no more than half of the available machines are shut down at any time. They prove that an upper bound for an LPT-schedule is 2, and that this bound is tight. The bound would be infinity if the machines were allowed to be shut down all at the same time, for any amount amount of time, which is why assumptions like the above are needed. We show that an asymptotic lower bound for any algorithm to solve this problem is also 2, by reducing 3PFB to finding a schedule with a bound less than two by a constant for the scheduling problem.

The following lemma restates 3PFB in a form more suitable for our proving the lower bound, and Theorem B.6 states the result.

Lemma B.5 (Variation of 3-Partition with fixed bottom elements)
Let $\{a_1, a_2, .., a_{3m}\}$, with the first m elements being the distinguished ones be an instance of 3-Partition with fixed bottom elements, where all elements are positive and non-zero. Let $n_i = a_i/a_1$. Then for any positive λ a solution of the 3-Partition with fixed bottom elements $\{n_1\lambda, n_2\lambda, \ldots, n_{3m}\lambda\}$ (with $n_1\lambda, n_2\lambda, \ldots, n_m\lambda$ distinguished elements) implies a solution of the initial instance.

Proof: By multiplying the elements allotted to each set by the factor a_1/λ, we get a solution of the initial problem. $\triangle$

Theorem B.6 (Asymptotic Result)

Solving scheduling with machine shutdowns, where at most half of the processors can be shut down simultaneously (SMS1/2), within a constant bound less than 2 as compared to the optimal schedule is NP-hard. The asymptotic result is that any scheduling algorithm will miss the

$$2 - \frac{20\epsilon}{1 + 8\epsilon}$$

bound for some problem instances, where ϵ can be arbitrarily small.

Proof: Let $\{a_1, a_2, .., a_{3m}\}$ with $a_1, a_2, .., a_m$ distinguished elements be an instance of 3-Partition with fixed bottom elements, and $\epsilon < 0.1$ be a given value that is arbitrarily close to 0.

We build an instance of the SMS1/2-problem, the solution of which, within a factor less than 2 compared to the optimal solution, would correspond to a solution of the 3-Partition with fixed bottom elements instance.

According to Lemma B.5 this problem is equivalent to any 3-Partition with fixed bottom elements-instance $\{n_1\lambda, n_2\lambda, \ldots, n_{3m}\lambda\}$ with the $n_i = a_i/a_1$ calculated as in the lemma, and with $\lambda = \epsilon/(3n_{max} + 1)$, where n_{max} is the maximum among the n_i's. The set size for this problem is $B = (\sum_{i=1}^{3m} n_i\lambda)/m$. Note that $B < \epsilon$, since $B \leq 3mn_{max}\frac{\lambda}{m} < 3\lambda n_{max} + \lambda = \epsilon$.

We are constructing now an instance of the SMS1/2-problem. The number of machines is $p := 2m$. The jobs are given as follows:

- $2m$ *big* jobs with processing time: $1/2 + 4\epsilon$
- $2m$ *normal* jobs with processing times: $n_i\lambda + \frac{1}{2} - \epsilon$ for $i \in \{m+1, m+2, \ldots, 3m\}$

The first m machines are shut down in the following intervals:

$$(1 - 2\epsilon - n_j\lambda + B, \frac{3}{2} - 3\epsilon]$$

for $j \in \{1, 2, \ldots m\}$. Recall that for this j-range the $n_j\lambda$ were the distinguished elements.

The other m machines shut down at times:

$$(\frac{3}{2} - 3\epsilon, \frac{3}{2}]$$

Note that this transformation can be done in polynomial time.

The following statement is of major importance to the proof:

(1) There is a solution of the 3PFB-instance if and only if there is a schedule of the SMS-instance that ends at time $1 + 8\epsilon$.

Given the 3PFB solution schedule for each set's composition (a_j, a_i, a_k) — we have $j \in [1..m]$ and $i, k \in [m + 1..3m]$ — the normal jobs of length $n_i\lambda + \frac{1}{2} - \epsilon$ and $n_k\lambda + \frac{1}{2} - \epsilon$ on the machine that shuts down at $1 - 2\epsilon - n_j\lambda + B$. This is possible since $n_i\lambda + n_k\lambda + n_j\lambda = B$. These schedules end before time $1 - 2\epsilon + B < 1 - \epsilon$. To each of the remaining machines we can assign two big jobs, thus ending the schedule at $1 + 8\epsilon$.

Now given a schedule ending at most at time $1 + 8\epsilon$ we show that there is a solution of 3PFB that can be derived from it.

Such a schedule must have all jobs scheduled before the machine shutdowns. Then we have:

($*$) There must be exactly two jobs on each processor.

($**$) No big job can be scheduled on the first m machines.

Proof of ($*$): All processing times are greater than $1/2-\epsilon$ thus no three jobs executed one after the other can be executed within $3/2-3\epsilon$ time, and this is more than $1+8\epsilon$ for small ϵ.

Due to the number of jobs there need to be exactly two jobs on each processor for the schedule to end at $1+8\epsilon$.

Proof of ($**$): Adding the least possible job to the processing time of a big job we get a quantity greater than $1-\epsilon$, which is greater than the start of any of the downtimes of the first m machines: $\frac{1}{2}+4\epsilon+\frac{1}{2}-\epsilon > 1+2\epsilon \geq 1-\epsilon-n_j+B$ for any j, since $B < \epsilon$.

From ($**$) and ($*$) we conclude that the given schedule must have two normal jobs on each of the first m machines and two big jobs on each remaining machine.

From the schedules on the first m machines we can find a solution of the 3PFB-instance, which completes the proof for (1).

Next, we show that the ratio between the length of the optimal schedule s_{opt} and the length of the next best schedule s can be arbitrarily close to 2. We proceed by proving the following statement:

(2) Finding a schedule that ends at $1+8\epsilon$ is NP-hard. Thus any polynomial algorithm will sometimes miss the solution, assuming $P \neq NP$. This follows from (1).

If the schedule does not end at $1+8\epsilon$, one job needs to be scheduled after the shutdown: no sum of two jobs is greater then $1+8\epsilon$, and no three jobs can fit in any pretime, as their sum is greater than $\frac{3}{2}-3\epsilon$, thus the late end of the schedule must come from a job being scheduled after a downtime. Its first possible start time is when the first processors wake up, i.e. $3/2 - 3\epsilon$. Then its finish time is greater than $s = 3/2 - 3\epsilon + 1/2 - \epsilon$, which is $2 - 4\epsilon$. The ratio is

$$\frac{s}{s_{opt}} \geq \frac{2-4\epsilon}{1+8\epsilon} = 2 - \frac{20\epsilon}{1+8\epsilon}.$$

△

C. Conclusion

In this chapter we have presented an LPT-based algorithm, the schedule of which ends within 3/2 of the time needed by the optimal schedule or of the end of the last downtime. This bound is tight in the class of polynomial algorithms assuming that $P \neq NP$. The difference between our algorithm and LPT is that it orders the processors before applying LPT in a way that facilitates the proof.

The proof of the upper-bound result based on the existence of a *compensating processor*, a processor that has more processing time in the optimal schedule than in the schedule of our algorithm.

A second result concerns the tightness in the class of polynomial algorithms assuming $P \neq NP$ of the bound of 2 when no more than half the machines shut down at the same time, which was obtained for LPT in [6].

Depending on the setting of the problem, i.e. which assumptions about the downtimes apply, our result or results in other papers provide more information about the worst-case bound of polynomial algorithms when scheduling in the presence of machine shutdowns (assuming that $P \neq NP$).

The LPTX-algorithm achieves best worst-case bounds in both considered situations.

CHAPTER III

SCHEDULING ON UNIFORM PROCESSORS WITH AT MOST ONE DOWNTIME ON EACH MACHINE

In this chapter we consider nonpreemptive scheduling on uniform processors each of which have at most one period of unavailability or at most one fixed job. In the next subsection we formally define the problem and introduce the FFDL Multifit scheduling algorithm, while subsection B contains the upper bound result.

A. Preliminary remarks

In this subsection we introduce the FFDL Multifit scheduling algorithm, and set the stage for showing that its schedule finishes within 3/2 of the optimal schedule's end or of the end of the last downtime. According to the tightness result in from subsection B of Chapter II, this bound is asymptotically tight in the class of polynomial algorithms assuming that $P \neq NP$.

Definition A.1 (Problem Instance)

A problem instance is given by a tuple $(P, T, \alpha : P \to Q, \gamma : P \to Q, \delta : P \to \mathbb{N}, d : T \to \mathbb{N})$, such that for all $p \in P$ we have $\alpha(p)\gamma(p) \in \mathbb{N}$. $\mathbb{N}$ represents the set of natural numbers, while Q is the set of rational numbers. Here,

- P is a set of processors,
- T is a set of tasks,
- $d(X)$ denotes the duration of a task X, as a number of time units the task needs to execute on the slowest processor, or of computing units.
- $\delta(p)$ denotes the start of the downtime of a processor p,
- $\gamma(p)$ denotes the end of the downtime of the processor p,

- $\alpha(p)$ is the speed factor of the processor p, meaning that the time a task X takes to execute on p is $\frac{d(X)}{\alpha(p)}$.

When processors do not have periods of unavailability, the multifit algorithm first assigns upper and lower bounds for the schedule lengths, and then proceeds with a binary search to find a schedule length that is within a desired degree of accuracy, using the FFD (first fit decreasing) algorithm to assign tasks to processors, each time a schedule length is considered. On uniform processors, the time slots to be filled with tasks can be ordered in increasing order of $\alpha(p)$ times the length of the time slot. The first fit decreasing (FFD) algorithm orders the tasks in decreasing order and then assigns each task to the first time slot encountered in which it fits. Our algorithm is:

```
Multifit($\epsilon$, upper bound, lower bound)
0) if (upper bound $-$ lower bound $< \epsilon$){
       print(upper bound);
       return;
   }
1)Set schedule length at $m = \frac{\text{upper bound} + \text{lower bound}}{2}$
2)for $p \in P$ {
       let $pre_p = \delta(p) * \alpha(p)$, and
       $post_p = (m - \gamma(p)) * \alpha(p)$;
       }
3) Order all time slots $pre_p$ and $post_p$ with $p \in P$ in increasing order of their length
and record then in an array TS[1..2$|P|$]
4) Order all tasks in decreasing order of their duration and record in an array $T[1..|T|]$
5) for $(i = 1; i \leq |T|; i++)$
```

```
        for $(j = 1, ; j \leq |TS|; j++)$
            if $(d(T[i]) \leq TS[j])$: {
                $TS[j] = TS[j] - d(T[i])$;
                $T[i] = nil$;
                }
6) if (all array entries $T[i]$ are $nil$)
        Multifit($\epsilon$, m, lower bound);
   else
        Multifit($\epsilon$, upper bound, m);
```

We call pretime of a processor p the length pre_p as defined above, and posttime the length $post_p$. They are the available times of a processor before its downtime starts and after its downtime ends, respectively. We also denote with $\gamma_p = \alpha(p)\gamma(p)$, and $\delta_p = \alpha(p)\delta(p)$.

We denote with $Multifit$ the end of the Multifit FFDL-schedule and with *opt* the maximum between end of the optimal schedule and the end of the last downtime: *opt* $= max(C_{max}(OPT), max_{p \in P}\gamma(p))$, where OPT is the optimal schedule.

B. Upper bound for FFDL Multifit schedule length

In this subsection we prove the following theorem:

Theorem B.1 (Bound for Multifit schedules)

The maximum completion time of a Multifit schedule is less or equal to 3/2 the maximum completion time of the optimal schedule or 3/2 the maximum end of a downtime,

$$Multifit < \frac{3}{2} max(C_{max}(OPT), max_{p \in P}(\gamma(p)) + \epsilon.$$

Here, $\epsilon > 0$ is the first input parameter of the Multifit algorithm.

We call a problem instance that fails to fulfill Theorem B.1 counterexample. We define a minimal counterexample, which is shown to exist whenever there is a counterexample. Then we prove several properties of a minimal counterexample, at last resulting in the fact that such a counterexample does not exist. Several theorems and lemmas contribute to this proof. We shall assume that the 3/2 bound is broken and derive a contradiction.

Definition B.2 (Order relation on problem instances)

Given two problem instances $C1 = (P1, T1, \alpha_1, \gamma_1, \delta_1, d_1)$, and $C2 = (P2, T2, \alpha_2, \gamma_2, \delta_2, d_2)$, we say that $C1 < C2$ if any of the following holds:

a) $|T1| < |T2|$

b) $|T1| = |T2|$ and $|P1| < |P2|$

c) $|T1| = |T2|$, $|P1| = |P2|$, and the number of processors with pretimes in $C1$ is less than the number of processors with pretimes in $C2$.

d) $|T1| = |T2|$, $|P1| = |P2|$, the number of processors with pretimes of both instances is the same, and, if $T1 = T2$, there is at least a task $X \in T1$ such that $d_1(X) < d_2(X)$, and $\forall X \in T1, d_1(X) \leq d_2(X)$. $|S|$ represents the number of elements in a set S.

Definition B.3 (Minimal Counterexample)

A minimal counterexample is a problem instance $C = (P, T, \alpha, \gamma, \delta, d)$, such that the Multifit schedule of C exceeds or is equal to $\frac{3}{2}opt + \epsilon$ and $\frac{3}{2}\underset{p \in P}{max}(\gamma(p)) + \epsilon$, where ϵ is the last input parameter of the Multifit FFDL algorithm, and such that C is minimal with regard to the order relation from Definition B.2.

Recall that $\epsilon > 0$ is a value chosen by the user, and thus, if it can be shown that there is no counterexample for any ϵ, then Multifit FFDL delivers a schedule that is within $\frac{3}{2}$ the optimal schedule or $\frac{3}{2}$ the end of the last downtime, if the value ϵ is chosen to be small enough.

Lemma B.4 (Existence of minimal counterexample)

If there is a counterexample then there also is a minimal counterexample.

Proof: Suppose we have a set S of counterexamples. Then there must be a subset S1 of counterexamples which have a minimum number of processors. Among those there must be a subset with a minimum number of tasks. Last there must be a subset of counterexamples of this last set that have a minimum number of processors with pretimes. Among these counterexamples, for each counterexample $C1 = (P1, T1, \alpha_1, \gamma_1, \delta_1, d_1)$, there is a finite set of counterexamples S(C1) that have tasks of lesser or equal durations to their task lengths, and the same task names. The counterexample $C2 = (P1, T1, \alpha_1, \gamma_1, \delta_1, d_2)$ with the least durations for each task name in $T1$, is the minimal counterexample corresponding to $C1$. Thus if there is a counterexample then there also is a minimal counterexample. $\triangle$

From now on we will assume that the counterexample we are considering is minimal. For convenience we will have the task names also represent their durations when this creates no ambiguity, for example if there is a task X its duration will be denoted by X.

Next we consider the FFDL-schedule of a minimal counterexample $(P, T, \alpha, \gamma, \delta, d)$ when the Multifit assigned deadline is set at $m \geq max(\frac{3}{2}C_{max}(OPT), max_{p \in P}(\gamma(p)))$. such that FFDL fails to return a successful schedule.

Let $\overline{X}$ be the first task that FFDL can not fit when m is set as a deadline. All tasks that are less than $\overline{X}$ FFDL would schedule after $\overline{X}$ are irrelevant to the fact that the FFDL schedule breaks the bound, thus a minimal counterexample only contains tasks that are greater or equal to $\overline{X}$. We state this in the following lemma.

Lemma B.5 (Length of tasks)

A minimal counterexample contains only tasks that are greater or equal to the first task that can not be scheduled by FFDL when the deadline assigned by Multifit is equal to or greater than $max(\frac{3}{2}C_{max}(OPT), max_{p \in P}(\gamma(p)))$.

Note that $\overline{X}$ does not belong to the FFDL-schedule of $(P, T, \alpha, \gamma, \delta, d)$, when the Multifit bar is set at m.

In the following we normalize everything to the length $\overline{X}$ of the task $\overline{X}$, that is, a number will represent that same number times the task length $\overline{X}$ in the measuring of time. We continue by showing some more properties.

Lemma B.6 ($\geq$ 1 tasks in FFDL pretimes)

In a minimal counterexample the FFDL-schedule has at least one task in the pretime of each processor that has a pretime.

Proof: If the pretime of a processor p is empty in the FFDL-schedule, then the last task $\overline{X}$ did not fit in that pretime. But $\overline{X}$ is the least task, and thus the optimal schedule could also fit nothing in that pretime. Then we can build a lesser counterexample by maintaining the same processors and tasks with the difference that the pretime of p is replaced by downtime. Both the optimal schedule and the FFDL schedule will remain the same, and the new counterexample has fewer pretimes. △

Notation B.7 (Lengths of Time)

In addition to normalizing every time length to the length of the task $\overline{X}$ we also use the following notations. Let p_* be the processor with the slowest speed.

- $opt_p = \alpha(p)opt$ for any $p \in P$
- $\gamma_p = \alpha(p)\gamma(p)$
- $\delta_p = \alpha(p)\delta(p)$
- Given a weight function $w : T \to R$, the task density of a task X is $\rho(X) = \frac{w(X)}{X}$. Here R represents the set of real numbers. Given a set of task types

$SQ = \{Q_1, Q_2, \ldots, Q_n\}$ with lengths $X_i \in [a_i, b_i)$ and weights $w(X_i) = a_i$ for all tasks X_i of type Q_i (or Q_i-tasks), we denote with $\rho_{Q_i} = \frac{a_i}{b_i}$. Note that

$$\rho_{Q_i} < \rho(X_i) \text{ for all } Q_i\text{-tasks } X_i \; .$$

Lemma B.8 (Task density)

To fill a number t of computation units (or a time t) with tasks of types belonging to a set SQ the FFDL Multifit algorithm needs to use tasks of total weight

$$w_{alg} > \underset{Q \in SQ}{min} \rho_Q * t.$$

Here we assume that $SQ = \{Q_1, Q_2, \ldots, Q_n\}$ with lengths $X_i \in [a_i, b_i)$ and weights $w(X_i) = a_i$ for all tasks Q_i-tasks X_i.

Proof: Since $\rho_{Q_i} < \rho(X_i)$ for all Q_i-tasks X_i, the total weight of a set of tasks that fill time t will be greater than $\underset{Q \in SQ}{min} \rho_Q * t$. △

Lemma B.9 (Idle times in FFDL schedules)

The idle time in the FFDL-schedule of the pretime or posttime of any processor is shorter than 1.

Proof: Suppose this is not the case and we have a processor p on which this situation is encountered. Suppose there is a time slot pre_p or $post_p$ with an idle time that is greater or equal to 1. Then FFDL would have scheduled $\overline{X}$ in that pretime or posttime and finished within 3/2 the end of the optimal schedule or 3/2 the last end of a downtime and we would not have a counterexample. △

To better describe schedules on processors we will use the same notation as in the previous chapter for processor schedules:[will denote start of the schedule, time 0, | will represent the downtime, and $[A_1 A_2 \ldots A_n | B_1 B_2 \ldots B_m$ will denote a schedule that has the tasks $A_1, A_2, \ldots A_n$ in the pretime in the given order and the tasks $B_1, B_2, \ldots B_m$ in the given order after the downtime.

We call Y-tasks all tasks the length of which is in the interval $[1.5, 2)$. Also, we denote with $busy_{ALG}(p)$ length of the total processing time of processor p in an ALG-schedule.

Theorem B.10 (Constraint for FFDL-schedule)

There is a processor p the FFDL-schedule of which contains a Y-task in its pretime and $pre_p \geq 2.5$ or in its posttime and $post_p \geq 2.5$. Recall that Y-tasks are tasks of the length $[1.5, 2)$.

Proof: We use a weighing argument. Consider the following task categories: $X \in [1, 2)$ with $w(X) = 1$ and $R_i \in [i, i+1)$ for $i \geq 2$ with $w(Z) = i$ for all R_i-tasks Z.

We consider the total weight of the tasks in the FFDL-schedule omitting $\overline{X}$ (which would not have fit if the Multifit bar was at $max(\frac{3}{2}opt, \frac{3}{2}max_{p \in P}(\gamma_p)))$, and compare it to the total weight of the tasks in the optimal schedule. We do this by considering each processor separately. For a given processor p we denote with $w_p(ALG)$ the sum of the weights of the tasks scheduled by the algorithm ALG on the processor p. We denote with OPT an optimal algorithm.

We show that $w_p(FFDL) \geq w_p(OPT)$ for each processor p which leads to a contradiction since we should have $w(FFDL) = w(OPT) - 1$, as the FFDL-schedule does not contain $\overline{X}$ when the Multifit assigned deadline is m. Denote with $w_{alg.after}(p)$ the total weight of the tasks scheduled by the $FFDL$ algorithm in the posttime of p, and with $w_{alg.pre}(p)$ the weight on the tasks scheduled by $FFDL$ in the pretime of p. Recall that we are considering the FFDL schedule when the Multifit assigned deadline is m, and $\overline{X}$ does not fit in this schedule. Let $w_{alg}(p) = w_{alg.after}(p) + w_{alg.pre}(p)$ be the weight of the $FFDL$-schedule in p. Suppose that the optimal schedule has a weight n in the pretime of p and weight m on the posttime of p. Then $w_{opt}(p) = n + m$.

First, we consider pretimes and posttimes that are greater or equal to 2.5. We assume that there are no Y-s in pretimes or posttimes that are greater or equal to 2.5, and thus the maximum task type density for the tasks we have in those times is $\frac{1}{1.5}$. From Lemma B.8 we have:

$$w_{alg.after}(p) > \frac{\frac{3}{2}m + \frac{n}{2} - 1}{1.5} = m + \frac{n-2}{3},$$

and thus $w_{alg.after}(p) \geq m + \lfloor\frac{n-2}{3}\rfloor + 1$, where $\lfloor x \rfloor$ denotes the highest integer value less or equal to x. Suppose there are no Y-tasks in the pretimes of processors that are greater or equal to 2.5. Suppose $p \in P$ and $pre_p \geq 2.5$, and $post_p \geq 2.5$. Then, again, $\underset{Q \in SQ \setminus \{Y\}}{min} \rho_Q = \frac{1}{1.5}$. Then we have

$$w_{alg.pre}(p) > \frac{n-1}{1.5} = \frac{2n-2}{3},$$

since the idle time in the pretime is less than 1, and by Lemma B.8. Thus $w_{alg.pre}(p) \geq \lfloor\frac{2n-2}{3}\rfloor + 1$. Replacing for $n \in \{3k, 3k+1, 3k+2\}$ we get $w_{alg}(p) = w_{alg.pre}(p) + w_{alg.pre}(p) \geq m + n \geq w_{opt}(p)$. For $n = 3k$, we have $w_{alg}(p) \geq m + \lfloor\frac{n-2}{3}\rfloor + 1 + \lfloor\frac{2n-2}{3}\rfloor + 1 = m + k - 1 + 1 + 2k - 1 + 1 = m + n$. For $n = 3k+1$ we have $w_{alg}(p) \geq m + \lfloor\frac{n-2}{3}\rfloor + 1 + \lfloor\frac{2n-2}{3}\rfloor + 1 = m + k - 1 + 1 + 2k + 1 = m + n$. For $n = 3k+2$ we have $w_{alg}(p) \geq m + \lfloor\frac{n-2}{3}\rfloor + 1 + \lfloor\frac{2n-2}{3}\rfloor + 1 = m + k + 1 + 2k + 1 = m + n$.

Next, we consider processors that have pretimes or posttimes that are less than 2.5. If $n = 0$, and $m \leq 1$ we have $w_{alg}(p) \geq w_{opt}(p)$, since FFDL needs to schedule a task different from $\overline{X}$ on p if $m = 1$ (since $post_p - 1 > 0$), and $w_{opt}(p) = 0$ if $m = 0$. If $m = 2$ we have $post_p \geq \frac{3}{2} * 2 = 3$, and so FFDL would need at least two X-tasks or one R_i-task to fill the time $post_p - 1$, and so $w_{alg}(p) \geq w_{opt}(p)$. For $m \geq 3$, by assumption there are no more Y-tasks, and so, as shown above,

$w_{alg.after}(p) \geq m + \lfloor \frac{n-2}{3} \rfloor + 1 = m$, and $w_{alg}(p) \geq w_{opt}(p)$.

Next, we consider $n = 1$. Since $\overline{X}$ would fit in this pretime (and not break the 3/2 bound), at least one task (different from $\overline{X}$) must be scheduled by FFDL in it. If $m = 0$, the inequality to prove holds. If $m = 1$, $w_{alg.after}(p) \geq 1$, and the inequality holds again. If $m \geq 2$, we have $post_p \geq \frac{3}{2}opt_p - \gamma_p \geq \frac{1}{2}\gamma_p + \frac{3}{2}m \geq 3.5$. Thus we have $w_{alg.after}(p) \geq m + \lfloor \frac{n-2}{3} \rfloor + 1 = m$, and $w_{alg}(p) \geq w_{opt}(p)$.

If $n = 2$, $w_{alg.pre}(p) \geq 1$, else $\overline{X}$ would fit in p's pretime. If $m = 0$, we have $w_{alg.after}(p) \geq 1$, and $w_{alg}(p) \geq w_{opt}(p)$. If $m \geq 1$, $post_p \geq \frac{3}{2}m + \frac{1}{2}n = 2.5$, and we have $w_{alg.after}(p) \geq m + \lfloor \frac{n-2}{3} \rfloor + 1 = m + 1$, and thus $w_{alg}(p) \geq m + 2 = w_{opt}(p)$. $\triangle$

Corollary B.11 (Long tasks in FFDL-pretimes)

In each FFDL-pretime or posttime that is less long than 2.5, but greater than than or equal to 2, there is at least one task the length of which is at least 1.5.

Definition B.12 (Compensating processor)

A processor that has less busy time in the FFDL-schedule, without considering the last task $\overline{X}$, than in the optimal schedule is called a compensating processor.

Lemma B.13 (Existence of compensating processors)

There is at least one compensating processor.

Proof: The total busy time of the FFDL-schedule without the last task $\overline{X}$ is less than the total busy time of the optimal schedule since the optimal schedule contains task $\overline{X}$. Thus there must be a processor on which OPT has more busy time than FFDL without task $\overline{X}$. $\triangle$

Theorem B.14 (Structure of a compensating processor)

Let p be a compensating processor. Then p has a pretime and:

(a) $opt_p < 4$. The total number of computation units (task units/time units on slowest processor) needed by the tasks in the optimal schedule on p is < 4

(b) the optimal schedule on p is of the following form: $[X_1X_2|X_3$, where X_1, X_2, and X_3 are arbitrary tasks, or $[Z_1|X_3$, where $Z_1 \geq 2$.
(c) the FFDL schedule on p is of the form $[Y|X_4X_5$, where Y is a Y-task, and $X_4, X_5 \in T$.

Proof: Suppose p has no pretime. If $busy_{OPT}(p) \geq 2$ then $busy_{FFDL}(p) > post_p - 1 \geq \frac{3}{2}busy_{OPT}(p) + \frac{1}{2}\gamma_p - 1 \geq busy_{OPT}(p) + \frac{(busy_{OPT}(p)-2)}{2} \geq busy_{OPT}(p)$, and thus p is not compensating.

Thus $busy_{OPT}(p) < 2$, and so the optimal schedule has only one task X_1 on p, and $X_1 < 2$. Since p is compensating, $busy_{FFDL}(p) < X_1 < 2$, and thus $FFDL$ also has only one task X_2 on p and $X_2 < X_1$. Let $X_{01}, X_{02}, ..., X_{0n}$ be all tasks in T such that $X_1 \geq X_{0i} > X_2$, for all $i \in \{1, 2, \ldots n\}$ and $X_{0i} \geq X_{0i+1}$ for $i \in \{1, 2, \ldots n-1\}$. With other words $X_1 \geq X_{01} \geq X_{02} \geq \cdots \geq X_{0n} > X_2$. Since X_2 was scheduled alone in a posttime that would have fit X_1 as well ($post_p \geq \frac{3}{2}X_1$), it results that X_1 was scheduled before the posttime $post_p$ was encountered, and must have been scheduled in a pretime or posttime $t_2 < post_p$. Since all tasks $X_{0i} \leq X_1$ and would also have fit in $post_p$, they also must have been scheduled in time slots $ts_{0i} \leq post_p$. Reducing the durations of $X_1, X_{01}, \ldots X_{0n}$ to the duration of X_2, we obtain a lesser counterexample. The FFDL schedule stays the same, since $\overline{X}$ and no other tasks would not fit in any of the increased spaces, and the order in which FFDL schedules tasks stays the same, while the optimal schedule can either improve or stay the same by scheduling the reduced tasks in the same places where they were scheduled in the optimal schedule of $(P, T, \alpha, \gamma, \delta, d)$. Thus p is not a part of a minimal counterexample if it is compensating, has no pretime, and $busy_{OPT}(p) < 2$. Thus p has a pretime.

Suppose $opt_p \geq 4$. Then $\frac{3}{2}opt_p - 1 - opt_p \geq \frac{1}{2}opt_p - 1 \geq 1$. By Lemma B.9, the idle time in the FFDL pretime of p is less than 1, and p is not compensating. Thus

$opt_p < 4$.

The FFDL schedule on p has more busy time than the length of the pretime of p, since the idle time in the pretime is less than 1, and FFDL must schedule a task after the downtime since $post_p - 1 \geq \frac{1}{2}\gamma_p - 1 > 0$. Thus the optimal schedule on p must have a task X_3 after the downtime.

Suppose the optimal schedule has only one task X_1 in the pretime of p. We know that $FFDL$ must contain at least a task X_4 in the pretime and a task X_5 after the downtime. Then we have $X_5 \geq \frac{1}{2}X_1 + \frac{3}{2}X_3 - 1 \geq X_3 + \frac{1}{2}X_1 + \frac{1}{2}X_3 - 1 \geq X_3$. For p to be compensating we need to have $X_1 > X_4$. If X_4 has not been scheduled alone in the pretime of p, then X_1 must exceed the length of two tasks for the processor to be compensating, $X_1 \geq 2$, and (b) holds. Suppose X_4 is the only task scheduled by FFDL in the pretime of p. Then FFDL scheduled X_4 in a pretime where X_1 would have fit. Thus X_1 was scheduled before that in a pretime or posttime that is less or equal to pre_p. Suppose $X_{01}, X_{02}, \ldots, X_{0n}$ are all tasks in T with $X_1 \geq X_{0i} > X_4$, and $X_1 \geq X_{01} \geq X_{02} \cdots \geq X_{0n} > X_4$. Reducing all tasks X_{0i} with $i \in \{1, \ldots, n\}$ to the duration of X_4 we get a lesser counterexample as the $FFDL$ schedule still breaks the $\frac{3}{2}$ bound.

Thus the optimal schedule has two tasks in the pretime and (b) follows.

From Corollary B.11, we know that the FFDL schedule on p has a task ≥ 1.5 in the pretime, if $pre_p < 2.5$. If $pre_p \geq 2.5$, then FFDL would need at least one Y-task, two tasks of length less than 1.5 to fill $pre_p - 1$, or a task that is greater or equal to 2.

The busy time after the downtime in the FFDL schedule is greater than $post_p - 1 = X_3 + \frac{1}{2}opt_p - 1 \geq 1.5$, since from Statement (b) we have $opt_p \geq 3$. If there were two tasks in the pretime of the FFDL-schedule, one of which is a Y-task, the busy time would be $\geq 1.5 + 1 + 1.5 = 4$ and p could not be compensating because

$opt_p < 4$, and we have (a). Thus there is only one task, Y_1, or two X-tasks during the pretime. Note that $post_p - pre_p = -\frac{1}{2}pre_p + \frac{3}{2}X_3 \geq -\frac{1}{2}pre_p + 1.5 > 0$ (because $pre_p + X_3 < 4$), and thus $pre_p < post_p$.

If there were only one task, Z, scheduled by FFDL after the downtime, then we have $Z > post_p - 1 \geq 1.5$, and there are two cases depending on whether Z would fit in the pretime. We consider case 1, when $Z \leq pre_p$. Then there can not have been two X-tasks in the pretime, since FFDL would have tried to schedule Z there first. So there is only one task Y_1 in the pretime, and $Y_1 \geq Z > X_3 + \frac{1}{2}opt_p - 1 \geq \frac{1}{2}opt_p$ and $Z + Y_1 > 2(\frac{1}{2}opt_p) = opt_p$ and p is not compensating. This argument also works if $Y_1 \geq 2$, since then Y_1 is still greater than Z. We consider case 2, when $Z > pre_p$. Then there can not have been two tasks scheduled by FFDL in the pretime of p, as then $busy_{OPT}(p) > busy_{FFDL}(p) > 4$. Thus FFDL scheduled only a task Y_1 in the pretime of p, and since $Z > X_1 + X_2$, $X_3 > Y_1 \geq 1.5$. Then $Z + Y_1 > X_3 + (\frac{3}{2}opt_p - 1 - opt_p) + Y_1 > X_3 + \frac{1}{2}opt_p - 1 + Y_1 \geq 2Y_1 - 1 + \frac{1}{2}opt_p \geq 2 + \frac{1}{2}opt_p > opt_p$ and again p is not compensating. Thus FFDL scheduled two tasks after the downtime of p.

Next we consider the size of Y_1. If $Y_1 \geq 2$, then $busy_{FFDL}(p) \geq 4$, and the optimal schedule can not be longer than that. Thus $Y_1 < 2$. This completes the proof of (c). $\triangle$

Lemma B.15 (Small tasks in some time slots)

If $pre_p \geq 3.5$, or $post_p \geq 3.5$ then any X-tasks scheduled by FFDL in pre_p or $post_p$ respectively have a length < 1.25. In particular, if the optimal schedule has tasks ≥ 1.5 in the posttime of p, and $pre_p \geq 2.5$, any X-tasks scheduled in the posttime of p are < 1.25. In this context, we consider X-tasks to be of length $[1, 1.5)$.

Proof: Consider a compensating processor cp, which exists by Theorem B.14. Note that $post_{cp} < 3.5$, else the tasks scheduled by FFDL on cp (which do not include $\overline{X}$) would add up to $2.5 + 1.5 = 4$, and since cp compensating, we would have $opt_p > 4$, which is not possible according to Theorem B.14. Thus the two X-tasks scheduled in $post_{cp}$ by FFDL are greater or equal to any X-tasks scheduled in $post_p$ if $post_p \geq 3.5$. Since the sum of the X-tasks scheduled by FFDL in $post_{cp}$ is less than 2.5 (else $busy_{OPT}(cp) > busy_{FFDL}(cp) \geq 4$), at least one of them is less than 1.25, and thus any tasks scheduled by FFDL in a pretime or posttime that is greater than 3.5 are less than 1.25.

Particularly, if $pre_p \geq 2.5$, and $post_p$ includes a task $Y \geq 1.5$ in the optimal schedule we have $post_p = \frac{3}{2}opt_p - \gamma_p \geq \frac{3}{2}(Y + \gamma_p) - \gamma_p \geq 2.25 + 1.25 = 3.5$ $\triangle$

Theorem B.16 (Constraint for FFDL-schedule)

There is a processor p with a pretime $2.5 \leq pre_p < 3$, the FFDL-schedule of which is $[Y_1|Y_2$ with both $Y_1 \geq 1.75$ and $Y_2 \geq 1.75$, and the optimal schedule of which is $[Y_3X_1|X_2$ or $[Z|X_4$, where $Z \geq 2.5$, $Y_3 \in [1.5, 2)$, and $X_1, X_2, X_3 \in [1, 1.5)$.

Proof: Suppose there is no such processor. We consider the following task types: X-tasks $\in [1, 1.5)$, Y-tasks $\in [1.5, 2)$, Z_1-tasks $\in [2, 2.5)$, Z_2-tasks $\in [2.5, 3)$, R_{11}-tasks $\in [3, 3.5)$, R_{12}-tasks $\in [3.5, 4)$ and R_i-tasks in the range $[i, i+1)$ for $i \geq 4$. For all task types Q that have tasks in the range $[a, b)$ let $w(Q_1) = a$ if Q_1 is a task of type Q. For a processor p let m be the weight of the optimal schedule after the downtime and n be the weight of the optimal schedule before the downtime. We use the notations $w_{alg.after}(p), w_{alg.pre}(p), w_{alg}(p), w_{opt}(p)$ from Theorem B.10.

We first consider the cases when $n < 3.5$. We show that inequality

$$w_{opt}(p) \leq w_{alg}(p) \qquad (1)$$

holds.

Suppose $n = 0$. If $m = 0$ or $m = 1$ (1) holds since in the second case $post_p - 1 \geq 0.5$ and $FFDL$ needs to schedule at least one task in $post_p$. If $m \geq 1.5$, $m < 2$, then the optimal schedule has only one task X_1 after the downtime of p. If $FFDL$ has two or more tasks in $post_p$, (1) holds. Suppose $FFDL$ has also scheduled only one task, X_2, in $post_p$, and $X_1 > X_2$. Then X_1 must have been scheduled in a time slot that is $\leq post_p$, and all tasks $X_{01}, X_{02}, \ldots, X_{0n}$ with $X_2 < X_{0i} \leq X_1$, and X_1, can be reduced to the length of X_2, creating a lesser counterexample, as the FFDL-schedule does not change, while the optimal can me at least maintained at the same length by putting the reduced tasks where they had been scheduled by OPT in the first place. So we have $X_2 \geq X_1$ and thus $w(X_2) \geq w(X_1)$, and (1) holds.

If $m = 2$, $post_p - 1 \geq \frac{3}{2}m - 1 = 2$, and $FFDL$ needs two tasks, or a task longer than 2 to fill this space. Thus (1) holds.

If $m = 2.5$, we have $post_p - 1 = 3.75 - 1 = 2.75$, and since 2 X-tasks or a Y-task, or a Z_1-task are not enough to fill this space, FFDL must have at least a Y-task and an X-task, or a Z_2-task or a bigger task, or 3 X-tasks in this space and $w_{alg}(p) \geq 2.5 \geq w_{opt}(p)$.

If $m = 3$, we have $post_p - 1 \geq 3.5$, and there are no more Y-tasks, and X-tasks are < 1.25. Thus there are at least 3 X-tasks or a R_{12}-task, or a R_{11} or a (Z_1 or a) Z_2-task and an X-task necessary to fill this space. (1) holds.

If $m \geq 3.5$ we have, since in this case all task densities are $> 1/1.25 = 4/5$, $w_{alg}(p) > \frac{4(\frac{3}{2}m-1)}{5} = \frac{6m-4}{5}$. Then $w_{alg}(p) \geq m$ if $m \geq 4$. Also $w_{alg}(p) \geq \lfloor \frac{6m-4}{5} \rfloor + 0.5 = \lfloor m \rfloor + \lfloor \frac{5(m-\lfloor m \rfloor)+m-4}{5} \rfloor + 0.5$. For $m = 3.5$ we get $w_{alg}(p) \geq 3.5 + \lfloor \frac{2.5+3.5-4}{5} \rfloor \geq w_{opt}(p)$. If $m > 3.5$, $m \geq 4$, and the statement to prove holds.

Suppose $n \in [1, 2)$. There can be only one task, X_1, in the pretime of the optimal schedule. $FFDL$ must also schedule a task X_2 in this pretime. If $X_1 > X_2$ a lesser counterexample can be created by reducing X_1 and all tasks greater than X_2 and $\leq X_1$ to the length of X_2. Since all these tasks have been scheduled by FFDL before X_2, they must have been assigned to smaller time slots, so the FFDL-schedule does not change. The optimal schedule stays the same or gets better, since all reduced tasks can be placed in their former places. Thus $X_2 \geq X_1$, and $w(X_2) \geq w(X_1)$. We have shown above that if $n = 0$ $w_{alg.after}(p) \geq m$ for all m.

Let $w_{alg.after}(p)(n_1, m_1)$ and $post_p(n_1, m_1)$ be the minimum total task weight and respectively the minimum $post_p$ a minimal counterexample can have after the downtime if $n = n_1$ and $m = m_1$. We have $post_p(q, m) > post_p(0, m)$ for all $q > 0$. Thus for $n \geq 1$ we have $w_{alg.after}(p)(n, m) \geq w_{alg.after}(p)(0, m) \geq m$. Thus $w_{alg}(p) = w_{alg.after}(p)(n, m) + w(X_2) \geq w_{alg.after}(p)(0, m) + w(X_1) \geq m + w(X_1) = w_{opt}(p)$.

Suppose $n = 2$. By Theorem B.10 there are still Y-tasks remaining when FFDL first reaches this pretime, and so the FFDL-schedule has a Y-task in the pretime of p. If $m = 0$, we have $post_p - 1 > \frac{3}{2}opt_p - 1 \geq 0$, and FFDL must have at least one task in $post_p$. If $m = 1$, $post_p \geq 2.5$, and FFDL must have at least one task of length ≥ 1.5 or two tasks in $post_p$. If $m = 2$, $post_p \geq 3 + 1 = 4$, thus FFDL must fill a space of at least 3, requiring at least an R_{11}- or greater task, a Z_1- or Z_2-task and at least an X-task, one Y-task and one X-task, two Y-tasks, or three X-tasks. Then $w_{alg}(p) \geq w_{opt}(p)$.

If $m \in \{2.5, 3\}$, $post_p \geq 4.75$, and a duration of 3.75 needs to be filled by the tasks assigned by the FFDL-schedule to $post_p$. By Lemma B.15 all X-tasks FFDL can have scheduled in $post_p$ are < 1.25, and there are no more Y-tasks. At least 4 X-tasks, one Z_1-task and two X-tasks, a Z_2-task or an R_{11}-task and one X-task, or an R_{12}- or greater task are needed to fill this time, and thus $w_{alg.after}(p) \geq 3.5$. Then $w_{alg}(p) \geq 5 \geq w_{opt}(p)$.

If $m \geq 3.5$, we have

$$w_{alg.after}(p) > \frac{4(\frac{3}{2}m + \frac{1}{2}n - 1)}{5} = \frac{6m + 4 - 4}{5} = \frac{6m}{5}$$

Thus $w_{alg.after}(p) \geq \lfloor \frac{m+5(m-\lfloor m \rfloor)}{5} \rfloor + m + 0.5$. If $m = \lfloor m \rfloor$ we have $w_{alg.after}(p) \geq m + 0.5 + \lfloor \frac{m}{5} \rfloor \geq m + 0.5$ and so $w_{alg}(p) \geq 1.5 + m + 0.5 = m + 2 = w_{opt}(p)$. If $m = \lfloor m \rfloor + 0.5$ we have $w_{alg.after}(p) = \lfloor m \rfloor + 0.5 + \lfloor \frac{m+2.5}{5} \rfloor \geq m + \lfloor \frac{3.5+2.5}{5} \rfloor = m + 1$. Then $w_{alg}(p) \geq m + 2.5 > w_{opt}(p)$.

Suppose $n = 2.5$. We have $w_{alg.pre}(p) \geq 1.5$, since at least a Y-task is needed to fill the time $pre_p - 1$. If $m = 0$, $post_p \geq 1.25$, so FFDL must have scheduled at least one task in $post_p$, and $w_{alg}(p) \geq 2.5 = w_{opt}(p)$.

If $m = 1$, we have $post_p \geq 1.25 + 1.5 = 2.75$. FFDL must have at least a Y-task Y_1, with $Y_1 \geq 1.75$, a task longer than 2, or two tasks in $post_p$. Suppose $pre_p \geq 3$. Then FFDL must have scheduled at least 2 X-tasks, a Y-task and an X-task, or a Z_1- or greater task in the pretime of p. Then $w_{alg}(p) \geq 2 + 1.5 = w_{opt}(p)$. We consider the case $pre < 3$. If $w_{alg.after}(p) \geq 2$, we have $w_{alg}(p) \geq 2 + 1.5 = w_{opt}(p)$. If not, then FFDL must have scheduled a Y-task, Y_0 in the pretime of p. Note that $post_p - pre_p \geq \frac{3}{2}m + \frac{1}{2}pre_p - pre_p = 1.5 - \frac{1}{2}pre_p > 0$, and $pre_p < post_p$. Thus Y_0 has been scheduled by FFDL before Y_1, and so $Y_0 \geq Y_1 \geq 1.75$. A processor with this FFDL-schedule has been outruled, however, at the beginning of this proof (since the weight of the pretime of the optimal schedule is 2.5 it must have a Y-task and an X-task or a Z_2-task scheduled there, and the only way to produce a weight of 1 in the posttime of p is to have an X-task in it).

If $m = 1.5$, $w_{opt}(p) = 4.5$. We have $post_p \geq 2.25 + 1.25 = 3.5$. By Lemma B.14 there are no more Y-tasks and all X-tasks are < 1.25. Also there are no more Z_1-tasks. Thus there are at least 3 X-tasks or an Z_2 or longer -task needed to fill in $post_p - 1$. Then $w_{alg}(p) \geq 1.5 + 2.5 = 4 = w_{opt}(p)$.

If $m = 2$, we have $post_p \geq 1.25 + 3$. To fill a time of 3.25 is the FFDL-schedule needs to have either 3 X-tasks, a Z_1 or Z_2-task and an X-task, or an R_{11}- or longer task in the posttime of p.

If $m = 2.5$, $w_{opt}(p) = 5$. $post_p \geq 3.75 + 1.25 = 5$. To fill a time of 4 FFDL needs at least 4 X-tasks that are < 1.25, a Z_2-task, R_{11} or R_{12} task and an X-task, or a task that has at least weight 4. Thus $w_{alg.after}(p) \geq 3.5$, and $w_{alg}(p) \geq 1.5 + 3.5 = 5 = w_{opt}(p)$.

If $m = 3$, $post_p \geq 1.25 + 4.5 = 5.75$. $m + n \geq 5.5$, To fill a time of 4.75 FFDL needs 4 X-tasks, a Z_2-task and two X-tasks, a R_{11} or R_{12} task and an X-task, or a task longer than 4 in the posttime of p. Then $w_{alg.after}(p) \geq 4$, and $w_{alg}(p) \geq 4 + 1.5 \geq 5.5 = w_{opt}(p)$.

If $m \geq 3.5$ we have

$$w_{alg.after}(p) > \frac{4(1.5m + 0.5n - 1)}{5} = \frac{6m + 2n - 4}{5} \geq \frac{6m + 1}{5}.$$

Then $w_{alg.after}(p) \geq \lfloor m \rfloor + 0.5 + \lfloor \frac{m+5(m-\lfloor m \rfloor)+1}{5} \rfloor$. If $\lfloor m \rfloor = m$, then also $m \geq 4$, and we have $w_{alg.after}(p) \geq m + 0.5 + \lfloor \frac{m+1}{5} \rfloor \geq m + 0.5 + 1$. If $\lfloor m \rfloor + 0.5 = m$ we have $w_{alg.after}(p) \geq m + \lfloor \frac{m+2.5+1}{5} \rfloor \underset{m \geq 3.5}{\geq} m + 1$ In both cases $w_{alg.after}(p) \geq m + 1$, and $w_{alg}(p) \geq m + 1 + 1.5 = w_{opt}(p)$.

Suppose $n = 3$. Then $w_{alg.pre}(p) \geq 2$, since an X-task alone or a Y-task alone in the pretime of p would leave an idle time that is ≥ 1. By always considering the minimum weight possible for $n = 2.5$ and all possible weight sums m that the optimal schedule can have scheduled after a downtime of a processor, we have shown above that for all m, $w_{alg.after}(p)(2.5, m) \geq m + 1$. We have $w_{alg.after}(p)(p) \geq w_{alg.after}(p)(3, m) \geq w_{alg.after}(p)(2.5, m) \geq m + 1$, and $w_{alg}(p) \geq w_{opt}(p)$ follows.

Suppose $n \geq 3.5$. We have, since the weights of all remaining tasks are less than $\frac{4}{5}$:

$$w_{alg.pre}(p) > \frac{4(n-1)}{5},$$

and $post_p \geq \frac{n}{2} \geq 1.75$. If $n = 3.5$, we have $w_{alg.pre}(p) > 2$, thus $w_{alg.pre}(p) \geq 2.5$. We have shown when considering the case $n = 2.5$ that $w_{alg.after}(p)(2.5, m) \geq m+1$. also $w_{alg.after}(p)(3.5, m) \geq w_{alg.after}(p)(2.5, m)$, and so $w_{alg}(p) \geq m+1+n-1 = m+n = w_{opt}(p)$.

Suppose $n = 4$. To fill a time of 3 FFDL needs to put at least 3 X-tasks, a Z_2-task and an X-task, or a task of type R_{11} or bigger in the pretime of p. Thus $w_{alg.pre}(p) \geq 3 = n-1$. We have $w_{alg.after}(p)(4, m) \geq w_{alg.after}(p)(2.5, m) \geq m+1$, and $w_{alg}(p) \geq m+1+n-1 = m+n = w_{opt}(p)$.

Suppose $n = 4.5$, and $\gamma_p < 5$. Then, as in the previous case, $w_{alg.pre}(p) \geq 3$. If $m = 0$, and $\gamma_p < 5$, we have $2.5 > post_p \geq 2.25$. Thus there is a Y-task in the posttime of p, and $w_{alg}(p) \geq 3 + 1.5 = 4.5 = w_{opt}(p)$. If $\gamma_p \geq 5$, then $post_p \geq 2.5$, and $w_{alg.pre}(p) \geq 1.5$, since a time of $post_p - 1 = 1.5$ can either be filled by a Y- or greater task or two X-tasks.

If $m = 1$, $post_p \geq 2.25 + 1.5 = 3.75$. When $post_p \geq 3.5$ we have:

$$w_{alg.pre}(p) > \frac{4(\frac{3}{2}m + \frac{1}{2}n - 1)}{5} = \frac{6m+2n-4}{5}, \text{ and}$$

$$w_{alg}(p) > \frac{4n-4}{5} + \frac{6m+2n-4}{5} = m+n+\frac{m+n-8}{5}.$$

Thus if $\frac{m+n-8}{5} \geq -0.5$, $w_{alg}(p) \geq m+n$. This is true because all weights are multiples of 0.5, so the weight of all tasks scheduled by FFDL on a processor p can not be in the interval $(m+n-0.5, m+n)$. We have $\frac{m+n-8}{5} \geq -0.5$ if $m+n \geq (-0.5)*5+8 = 8-2.5 = 5.5$. When $n = 4.5$, we have $m+n \geq 5.5$ whenever $m \geq 1$.

Suppose $n = 5$. Then we have $w_{alg.pre}(p) > (4pre_p - 4)/5 = 3.2$. Thus $w_{alg.pre}(p) \geq 3.5$. If $m = 0$ we have $post_p \geq 2.5$, and so a task of at least length 1.5, or two tasks are scheduled by FFDL in the posttime of p. We have $w_{alg.after}(p) \geq 1.5$ and $w_{alg}(p) \geq 1.5 + 3.5 = 5 = w_{opt}(p)$.

If $n \geq 5.5$ we have $m + n \geq 5.5$, and $w_{alg}(p) \geq w_{opt}(p)$, as shown above.

We have shown above that for any processor p we have $w_{alg}(p) \geq w_{opt}(p)$, a contradiction since $\sum_{p \in p} w_{alg}(p) = \sum_{p \in P} w_{opt}(p) - 1$, since the task $\overline{X}$ is part of the optimal schedule but not part of the $FFDL$ schedule. △

Lemma B.17 (Compensating processor)

(a) The posttime of any compensating processor *cp* is < 3.25.

(b) At least one task in its posttime is < 1.125.

Any pretimes or posttimes $pre_p \geq 3.25$ or $post_p \geq 3.25$ have an FFDL-schedule with the following properties:

(c1) It does not contain any Y-tasks.

(c2) Any X-tasks it contains are < 1.125.

(c3) It does not contain any tasks with the length in the interval $[2, 2.5)$.

Proof: Let *cp* be a compensating processor, which exists by Lemma B.13. Let $[Y_{cp}|X_1X_2$. be the schedule of *cp*. By Theorem B.16 we have a processor p with an $FFDL$ schedule of $[Y_0|Y_2$, with $Y_0 \geq 1.75$, and $Y_2 \geq 1.75$, and an optimal schedule of $Y_4X_3|X_4$. We have $post_p \geq \frac{1}{2} * 2.5 + 1.5 = 2.75$. Suppose $pre_{cp} \geq 2.75$. Then, since *cp* can only have an idle time of less than 1 in the pretime we have $Y_{cp} \geq 1.75$. If $pre_{cp} < 2.75$, then the assignment of Y_{cp} by FFDL happened before the assignment of Y_2 in $post_p$, and thus $Y_{cp} \geq Y_2 \geq 1.75$. Thus $Y_{cp} \geq 1.75$. Since $4 > bust_{OPT}(cp) > busy_{FFDL-\overline{X}}(cp) = Y_{cp} + X_1 + X_2$, we have $X_1 + X_2 < 4 - 1.75 = 2.25$. Thus $X_2 < 1.125$, since it was scheduled after X_1, and so $X_2 \leq X_1$ (and because their average is < 2.25). We have (b). Also since $X_1 + X_2$ must fill a time of at least

$post_{cp} - 1$ we must have $post_{cp} < 3.25$. We have (a). (c1) and (c2) follow from the fact that all pretimes or posttimes that are greater or equal to 3.25 are considered by FFDL after the posttime of cp.

Let p be the processor that was proved to exist in Theorem B.16. Its optimal schedule had a Y-task and an X-task in the pretime of p. Thus $pre_P \geq 2.5$. FFDL scheduled a Y-task in pre_p. Thus at the time it encountered pre_p there were no more tasks of length in the interval $[2, 2.5)$ available. Thus no tasks of that type can be scheduled in a pretime or posttime of length ≥ 3.25, and (c3) holds. $\triangle$

We continue by finishing the proof of Theorem B.1.

Proof: We consider the following task types and weights: X-tasks $\in [1, 1.5)$, Y_1-tasks $\in [1.5, 1.75)$, Y_2-tasks $\in [1.75, 2)$, Z_1-tasks $\in [2, 2.5)$, Z_{21}-tasks $\in [2.5, 2.75)$, and Z_{22}-tasks $\in [2.75, 3)$, R_{11}-tasks $\in [3, 3.5)$, R_{12}-tasks $\in [3.5, 4)$, R_{41}-tasks $\in [4, 4.5)$, R_{42}-tasks $\in [4.5, 5)$, and R_i-tasks $\in [i, i+1)$ for $i \geq 5, i \in \mathbb{N}$. The weights of each task type are equal to the smallest length a task of that type can have, for example if Y is of type Y_2, then $w(Y) = 1.75$.

Note that all weights are the same as those in the proof of Theorem B.16 with the sole exceptions of the weights of Y_2-tasks, Z_{22}-tasks, and R_{42}-tasks, all of which are greater than the weights tasks of their type had in that proof.

The FFDL-schedule of a minimal counterexample $(P, T, \alpha, \gamma, \delta, d)$ will not include $\overline{X}$ if the bar that defines the scheduling times after the downtime is at $max(\frac{3}{2}opt, max_{p \in P}(\frac{3}{2}\gamma(p)))$. Therefore we have $\sum_{p \in P} w_{FFDL}(p) < \sum_{p \in P} w_{OPT}(p)$. We next prove that for any $p \in P$ we have

$$w_{alg}(p) \geq w_{opt}(p) \qquad (1)$$

deriving a contradiction to the previous statement.

Since the new weights of the task types Y_2, Z_{22}, and R_{42} are greater than the weights the tasks those types had in the proof of Theorem B.16, all arguments regarding minimal weight of tasks in FFDL pretimes and aftertimes remain valid.

Thus, in all cases considered in Theorem B.16 where we had $w_{alg}(p) \geq w_{opt}(p)$ this still holds true.

The one case considered in the proof of Theorem B.16 where this did not hold was the processor p with $n = 2.5$ and $m = 1$, with a pretime $pre_p < 3$, for which the FFDL-schedule was $[Y_3|Y_4$ with both $2 > Y_3 \geq 1.75$, and $2 > Y_4 \geq 1.75$. With the new weights we have $w_{alg}(p) = 3.5 = w_{opt}(p)$, so inequality (1) holds in this case as well.

The new weights introduce the following new cases:

- $n \in \{1.75, 2.75, 3.25, 3.75, i+0.25, i+0.75 | i \in \mathbb{N} \text{ and } i \geq 4\}$, where $\mathbb{N}$ represents the natural numbers, and
- $m \in \{1.75, 2.75, 3.25, 3.75, i + 0.25, i + 0.75 | i \in \mathbb{N} \text{ and } i \geq 4\}$.

We first handle the cases when $n < 3.5$.

Suppose $n = 0$. If $m = 1.75$, we must have $OPT = [|X_1$ for some task X_1. If FFDL scheduled two or more tasks on p, we have (1). If it scheduled only one task X_2 on p we can use the same argument as in the proof of Theorem B.16 to show that is $X_2 \geq X_1$. Thus (1) holds.

If $m = 2.75$, we have $post_p \geq 4.125$. By Lemma B.17 there are no more Y-tasks or Z_1-tasks, and so to fill a time of 3.125 either 3 X-tasks or a Z_2 task and an X-task or a R_{11} or longer task are needed. We have $w_{alg}(p) \geq 3$.

The task density of all tasks scheduled by FFDL in time slots ≥ 3.25 is $> 5/6$, which results from Lemma B.17(c1), (c2) and (c3).

If $m \geq 3.25$ we have

$$w_{alg}(p) > \frac{5(\frac{3}{2}m - 1)}{6} = \frac{15m - 5}{12} = m + \frac{3m - 5}{12} \geq m + 0.4$$

Thus $w_{alg.after}(p) > m = w_{opt}(p)$.

Suppose $n \in [1, 2)$. The same argument as in the proof of Theorem B.16 holds: the FFDL pretime will include a task that is greater or equal to the task in the optimal pretime, thus $w_{alg.pre}(p) \geq n$, and since $post_p(0, m) < post_p(n, m)$ if $n > 0$ we have $w_{alg}(p)(n, m) \geq w_{alg}(p)(0, m) + w_{alg.pre}(p) \geq w_{alg}(p)(0, m) + n \geq m + n = w_{opt}(p)$.

Suppose $n = 2$. If $pre_p \geq 2.5$ we could have $n = 2.5$, which is handled in the next case. We assume $pre_p < 2.5$ We have $w_{alg.pre}(p) \geq 1.75$, since the processor proved to exist in Theorem B.16 had a Y_2-task in a pretime that is ≥ 2.5. If $m = 1.75$, $post_p \geq 3.625$. To fill a time ≥ 2.625 either 3 X-tasks, or a Z_2 or greater task are needed. We have $w_{alg}(p) \geq 2.5 + 1.75 > w_{opt}(p)$.

If $m = 2.75$, $post_P \geq 5.125$. At least a task greater than R_{11}, or an R_{11} or R_{12} and one X-task or Z_2-task and more than X-task, or 4 X-tasks, or a task of type R_{12} or greater are needed to fill a time of 4.125. Thus $w_{alg.after}(p) \geq 3.5$, and $w_{alg}(p) \geq 3.5 + 1.75 = 5.25 > w_{opt}(p)$

When $m \geq 3.25$ we have

$$w_{alg.after}(p) > \frac{5(\frac{3}{2}m + \frac{1}{2}n - 1)}{6} = \frac{15m}{12} = m + \frac{3m}{12} \geq m + 0.81$$

Thus, since both $w_{alg.after}(p)$ and the weight m must be multiples of 0.25 we have $w_{alg.after}(p) \geq m + 1$, and $w_{alg}(p) > m + 1 + n - 1 = w_{opt}(p)$.

Suppose $n \leq 2.5$, and $pre_p \geq 2.5$. We have $w_{alg.pre}(p) \geq 1.5$.

If $m = 1.75$, $post_p = 3.875$. To fill a time of 2.875 We know that for a compensating processor cp, $post_{cp} \geq 2.875$, since $opt \geq busy_{FFDL}(cp) \geq 3.75$, since FFDL had a Y_2-task in the pretime, and $\frac{3}{2}opt - opt + X_3 \geq 2.875$, where X_3 is the task scheduled by the optimal schedule after the downtime. Thus no tasks less than 2.875 are scheduled in time slots greater that $post_{cp}$. Need Z_{22}-tasks in the range $[2.75, 3)$ of weight 2.75.

To fill a time of length 2.875 either 3 X-tasks or a task of type Z_{22} or higher is needed, and $w_{alg}(p) \geq w_{opt}(p)$.

If $m = 2.75$, $post_p \geq 5.375$. Since all task densities are greater than $\frac{5}{6}$ we have $w_{alg.after}(p) > \frac{5}{6} * 4.375 > 3.645$. Thus $w_{alg.after}(p) \geq 3.75$, and $w_{alg}(p) \geq w_{opt}(p)$.

If $m \geq 3.25$ we have $w_{alg.after}(p) > \frac{5}{6}(\frac{3}{2}m - 1 + 1.25) = \frac{5}{6}(\frac{3}{2}m + 0.25) = \frac{15m+0.5}{12}$, and

$$w_{alg.after}(p) > m + \frac{3m + 0.5}{12} \geq m + \frac{9.75 + 0.5}{12} > m + 0.854,$$

and thus $w_{alg.after}(p) \geq m + 1$, and $w_{alg}(p) \geq m + n = w_{opt}(p)$.

Suppose $n = 2.75$. Since the idle time in the pretime is less than 1 there must be at least a task of type Y_2 or bigger, or two X-tasks in the pretime of p, and $w_{alg.pre}(p) \geq 1.75$. We have shown in the previous case that $w_{alg.after}(p)(2.5, m) \geq m + 1$. We have $w_{alg.after}(p)(2.75, m) \geq w_{alg.after}(p)(2.5, m) \geq m + 1$, and $w_{alg}(p) \geq m + n = w_{opt}(p)$.

Suppose $n = 3$. No Y-task by itself can fill a time of 2, so either a task of type Z_1 or bigger or two tasks are scheduled by FFDL in the pretime of p. thus $w_{alg.pre}(p) \geq 2$. Since $post_p(3, m) > post_p(2.5, m)$, we have $w_{alg.after}(p)(3, m) \geq w_{alg.after}(p)(2.5, m) \geq m + 1$, and $w_{alg}(p) \geq m + n = w_{opt}(p)$.

Suppose $n = 3.25$. By Lemma B.17, all X-tasks in this pretime have a length that is less than 1.125, and thus 2 X-tasks can not fill a length of 2.25. Also, by the same lemma, there are no Z_1-tasks in this pretime. Thus either 3 X-tasks, a Y-task and an X-task, or a task of type Z_2 or greater is needed to fill a space of 2.25, and $w_{alg.pre}(p) \geq 2.5 > n - 1$.

Then $w_{alg.pre}(p) \geq 2.5 > n-1$. Since $post_p(3.25, m) > post_p(2.5, m)$, we have $w_{alg.after}(p)(3.25, m) \geq w_{alg.after}(p)(2.5, m) \geq m+1$, and $w_{alg}(p) \geq m+n = w_{opt}(p)$.

If $n = 3.5$. We already had in the previous case $w_{alg.pre}(p) \geq 2.5 = n-1$. $w_{alg}(3.5, m) \geq w_{alg}(3.25, m) \geq 2.5 + m + 1 = m + n = w_{opt}(p)$.

Suppose $n \geq 3.75$. For all tasks Q that can be scheduled by FFDL in the pretime of p we have $\rho_Q < \frac{5}{6}$. Thus $w_{alg.pre}(p) \geq \frac{5(n-1)}{6}$. If $m \geq 1.75$ we have $post_p \geq 2.625 + 1.75 = 4.375 > 3.25$. Thus all task densities in the posttimes are also $< \frac{5}{6}$. We have: $w_{alg.after}(p) > \frac{5(\frac{3}{2}m+\frac{1}{2}n-1)}{6} = \frac{15m+5n-10}{12} = m + \frac{3m+5n-10}{12}$.
Thus:

$$w_{alg}(p) > m + \frac{10n-10+3m+5n-10}{12} = m + n + \frac{3n+3m-20}{12} \tag{2}$$

Thus if $\frac{3(m+n)-20}{12} > -0.25$, or $m + n \geq 5.67$, we have $w_{alg}(p) \geq w_{opt}(p)$. This is because all weights are multiples of 0.25, and thus if $w_{alg}(p) > m + n - 0.25$, then $w_{alg}(p) \geq m + n$.

If $n = 3.75$, we have: Since there are no more Y- or Z_1-tasks the FFDL-schedule must contain either 3 X-tasks, a task of type Z_{21} and at least one other task, or a task of type Z_{22} or a longer task in the pretime of p. $w_{alg.pre}(p) \geq 2.75 \geq n-1$. Because $w_{alg.after}(p)(3.75, m) \geq w_{alg.after}(p)(2.5, m) \geq m+1$, we have $w_{alg}(p) \geq n - 1 + m + 1 = n + m = w_{opt}(p)$.

If $m \geq 3.25$ we have from inequality (2) that $w_{alg}(p) > m + n$.

Suppose $n = 4$. $busy_{alg.after}(p) \geq 3$. Thus the FFDL-schedule contains at least 3 X-tasks, one Z_2-task and at least one more task, more than one Z_2-task, or a task of length 3 or greater in the pretime of p. $w_{alg.pre}(p) \geq 3 = n - 1$. Because $w_{alg.after}(p)(4, m) \geq w_{alg.after}(p)(2.5, m) \geq m+1$, we have $w_{alg}(p) \geq n-1+m+1 = n + m = w_{opt}(p)$.

Suppose $n = 4.25$, $pre_p < 4.5$ As in the case $n = 4$, we have $w_{alg.pre}(p) \geq 3$.

If $m = 0$, $2.25 \geq post_p \geq 2.125$, and $w_{alg.after}(p) \geq 1.5$, since Y-tasks are still available when the time slot $post_p$ is encountered by FFDL.

If $m = 1$, $post_p \geq 1.5 + 2.125 = 3.625$. To fill a time of 2.625 when no Y- or Z_1-tasks are available, at least 3 X-tasks, or a task of type Z_2 or longer must be used by FFDL in the posttime of p. $w_{alg.pre}(p) \geq 2.5 \geq m + 1.5$, and $w_{alg}(p) \geq w_{opt}(p)$.

if $m \geq 1.5$ we have $m + n \geq 5.75$, and thus $w_{alg}(p) \geq w_{opt}(p)$.

Suppose $n \in \{4.25, 4.5\}$, $pre_p \geq 4.5$. This case has been handled in Theorem B.16, when we considered $n = 4.5$ and assumed the pretime also to be ≥ 4.5. *to restate this* for all $m \leq 1.5$. If $m \geq 1.5$ we have $m+n \geq 5.75$, and $w_{alg}(p) \geq w_{opt}(p)$.

If $n = 4.75$, the FFDL-schedule must have either 4 X-tasks, a Z_{21}-task and at least two X-tasks or another Z_2-task, Z_{22}-task or an R_{11}-task and at least an X-task, or at least a task of type R_{12} or longer. Thus $w_{alg.after}(p) \geq 3.5$. We have shown that $w_{alg}(4.25, m) \geq m + 1.25$, thus $w_{alg}(4.75, m) \geq m + 1.25$, and $w_{alg}(p) \geq m + 1.25 + 3.5 = m + 4.75 = w_{opt}(p)$.

Suppose $n \in \{5, 5.25\}$. If $m > 0$, then $n + m \geq 6$, and $w_{alg}(p) \geq w_{opt}(p)$. If $m = 0$, $post_p \geq 2.5$, and $w_{alg.after}(p) \geq 1.5$. The FFDL-schedule must contain either 4 X-tasks, a Z_{21}-task and at least 2 X-tasks or another Z_2-task, a Z_{22}-task or an R_{11}- or R_{12}-task and at least an X-task, or an R_{41}- or longer task in the pretime of p. Thus $w_{alg.pre}(p) \geq 3.75$. We have $w_{alg}(p) \geq 5.25 \geq w_{opt}(p)$.

Suppose $n = 5.5$. If $m > 0$, then $n + m \geq 6$, and $w_{alg}(p) \geq w_{opt}(p)$. If $m = 0$, $post_p \geq 2.75$, and $w_{alg.after}(p) \geq 1.75$. We had in the previous case $w_{alg.pre}(p)(5, 0) \geq 3.75$. Thus also $w_{alg.pre}(p)(5.5, 0) \geq 3.75$, since $w_{alg.pre}(p)(5.5, 0) \geq w_{alg.pre}(p)(5, 0)$. Thus $w_{alg}(p) \geq 1.75 + 3.75 = 5.5 = w_{opt}(p)$.

If $n \geq 5.75$, we have $m + n \geq 5.75$, and the inequality to prove holds. $\triangle$

CHAPTER IV

SCHEDULING ON SAME-SPEED PROCESSORS WITH MULTIPLE SHUTDOWNS ON EACH MACHINE

In this chapter we consider scheduling on same-speed processors with possibly multiple downtimes on each machine. We adapt the FFDL Multifit scheduling algorithm from the previous chapter to this situation, and show that it finishes within $1.5 + \frac{1}{2k}$ multiplied by the end of the optimal schedule or by the end of the last downtime. We also show that the algorithm finishes within 3/2 the optimal maximum completion time or 3/2 the end of the last downtime when there are are most 2 downtimes on each machine.

The next subsection contains definitions and other preliminary considerations, subsection B contains the proof for the upper bound result, while subsection C contains the proof of an upper bound of 3/2, which is asymptotically tight in the class of polynomial algorithms assuming that $P \neq NP$, for the case when there are at most two downtimes on each machine.

A. Preliminaries

In this subsection we define a problem instance, describe the algorithm we use, and prove some theorems and lemmas which will be used in later subsections.

Definition A.1 (Problem Instance)

A problem instance is given by a tuple $(P, T, k \in \mathbb{N}, \delta : [1..k] \times P \rightarrow N, \gamma : [1..k] \times P \rightarrow \mathbb{N}, d : T \rightarrow \mathbb{N})$. Here, $\mathbb{N}$ represents the set of natural numbers, and

- P is a set of processors,
- T is a set of tasks,

- k is the maximum number of downtimes of the problem instance,
- $d(X)$ denotes the duration of a task X, as a number of time units the task needs to execute, or of computing units,
- $\delta(i,p)$ denotes the start of the i^{th} downtime of processor p if it exists, and is 0 otherwise,
- $\gamma(i,p)$ denotes the end of the i^{th} downtime of processor p if it exists, and is 0 otherwise.

Let γ_p denote the end of the last downtime of processor p and be 0 if there is no downtime on p. Also, let pre_{ip} denote the time slot starting at $\gamma(i-1,p)$, and ending at $\delta(i,p)$, where $\gamma(0,p)$ is defined to be 0.

When processors do not have periods of unavailability, the multifit algorithm first assigns upper and lower bounds for the schedule lengths, and then proceeds with a binary search to find a schedule length that is within a desired degree of accuracy, using the FFD (first fit decreasing) algorithm to assign tasks to processors each time a schedule length is considered. On processors with downtimes, the time slots to be filled with tasks can be ordered in increasing order of the length of the time slot. The first fit decreasing (FFD) algorithm orders the tasks in decreasing order and then assigns any task that fits to the first time slot encountered. Our algorithm is:

```
Multifit($\epsilon$, upper bound, lower bound)
0) if (upper bound $-$ lower bound $< \epsilon$){
        print(upper bound);
        return;
   }
1)Set schedule length at $m = \frac{\text{upper bound + lower bound}}{2}$
```

```
2) Order all time slots $pre_{ip} = \delta(i,p) - \gamma(i-1,p)$, and $post_p = m - \gamma_p$ with $p \in P$ in
increasing order of their length and record them in an array TS[1..$|TS|$], and discard
the empty ones,
3) Order all tasks in decreasing order of their duration and record in an array $T[1..|T|]$
4) for $(i = 1; i \leq |T|; i++)$
        for $(j = 1, ; j \leq |TS|; j++)$
            if $(d(T[i]) \leq TS[j])$: {
                $TS[j] = TS[j] - d(T[i])$;
                $T[i] = nil$;
                }
5) if (all array entries $T[i]$ are $nil$)
        Multifit($\epsilon$, m, lower bound);
   else
        Multifit($\epsilon$, upper bound, m);
```

We call posttime of a processor p the time length $post_p$. It is the available time of a processor after its last downtime ends, given a set Multifit schedule length.

Most of the remainder of this subsection is devoted to proving the following theorem, for problem instances with $k \leq 2$.

Theorem A.2 (Bound for Multifit schedules)

The maximum completion time of a Multifit schedule is less or equal to 3/2 the maximum completion time of the optimal schedule or 3/2 the maximum end of a downtime,

$$C_{max}(Multifit) \leq \frac{3}{2} max(C_{max}(OPT), max_{p \in P}(\gamma_p)) + \epsilon,$$

if the maximum number of downtimes in the problem instance is 2. Else

$$C_{max}(Multifit) \leq (\frac{3}{2} + \frac{1}{2k})max(C_{max}(OPT), max_{p\in P}(\gamma_p)) + \epsilon,$$

where k is the maximum number of downtimes of the considered problem instance. Here OPT denotes the optimal schedule, and ϵ the user-defined parameter of FFDL Multifit.

Note that, given that all time lengths are given as integer multiples of one unit, choosing ϵ to be less than half that unit will lead, if the theorem holds, to a schedule which ends within $1.5opt$ if there are at most two downtimes on one machine, and $(1.5 + \frac{1}{2k})opt$ if there are more than two downtimes on at least one machine. This is because the schedule ends at a time moment denoted by an integer and if the upper and lower bound are less far away than half a unit the upper bound can not be a unit above a schedule that ends within $3/2opt$, and so any schedule the upper bound produced ends within $3/2opt$. Here, $opt = max(C_{max}(OPT), max_{p\in P}\gamma_p)$.

We call a counterexample any problem instance the FFDL Multifit schedule of which ends after $\frac{3}{2}opt$. Next, we define a minimal counterexample, which is shown to exist whenever there is a counterexample. Then we prove several properties of a minimal counterexample, at last resulting in the fact that such a counterexample does not exist. Several Theorems and Lemmas contribute to this proof.

We denote the last task scheduled by the Multifit schedule with $\overline{X}$. In a minimal counterexample this is the only task which can not be scheduled when the multifit bar is at a time $b \geq 3/2(max_{p\in P}\gamma_p, C_{max}(OPT))$.

We denote with $FFDL$ the end of the FFDL-schedule and with opt the maximum between the end of the optimal schedule and the end of the last downtime.

In the following we shall assume that the 3/2 bound is broken and derive a contradiction for the case when there are maximum 2 downtimes, and prove that $FFDL < (1.5 + \frac{1}{2k})opt$ if the maximum number of downtimes on one processor (k) is greater or equal to 3.

Definition A.3 (Order relation on problem instances)

Given two problem instances $C1 = (P1, T1, k_1, \delta_1, \gamma_1, d_1)$, and $C2 = (P2, T2, k_2, \delta_2, \gamma_2, d_2)$ where T1 and T2 are sets of tasks with their execution times and P1 and P2 sets of processors with downtimes we say that $C1 < C2$ if any of the following holds:

a) $k_1 < k_2$ b) $k_1 = k_2$ and $|T1| < |T2|$

c) $k_1 = k_2$, $|T1| = |T2|$ and $|P1| < |P2|$

d) $k_1 = k_2$, $|T1| = |T2|$, $|P1| = |P2|$, and the number of processors with pretimes in $C1$ is less than the number of processors with pretimes in $C2$.

e)$k_1 = k_2$, $|T1| = |T2|$, $|P1| = |P2|$, the number of processors with pretimes of both instances is the same, and, if $T1 = T2$, there is at least a task $X \in T1$ such that $d_1(X) < d_2(X)$, and $\forall X \in T1, d_1(X) \leq d_2(X)$.

Definition A.4 (Minimal Counterexample)

A minimal counterexample is a problem instance $C = (P, T, k, \delta, \gamma, d)$, where such that the Multifit schedule exceeds $\frac{3}{2}C_{max}(OPT)$ and $\frac{3}{2}\max_{p \in P}(\gamma_p)$, such that C is minimal with regard to the order relation defined in Definition A.3. Recall that γ_p denotes the end of the downtime of processor $p \in P$.

We shall denote with *opt* the maximum between $C_{max}(OPT)$ and $max_{p \in P}(\gamma_p)$.

Lemma A.5 (Existence of minimal counterexample)

If there is a counterexample then there also is a minimal counterexample.

Proof: Suppose we have a set S of counterexamples. Then there must be a subset S1 of counterexamples which have a minimum k. Among these there must be a subset which have a minimum number of processors. Among those there must be a subset with a minimum number of tasks. Last there must be a subset of counterexamples of this last set that have a minimum number of processors with pretimes. Among these counterexamples, for each counterexample $C1 = (P1, T1, k_1, \delta_1, \gamma_1, d_1)$, there is a finite set of counterexamples S(C1) that have tasks of lesser or equal durations to their task lengths, and the same task names. The counterexample $C1 = (P1, T1, k_1, \delta_1, \gamma_1, d_2)$, with the least durations for each task name in $T1$, is the minimal counterexample corresponding to $C1$. Thus if there is a counterexample then there also is a minimal counterexample. $\triangle$

From now on we will assume that the counterexample we are considering is minimal. For convenience we will have the task names also represent their durations when this creates no ambiguity, for example if there is a task X its duration will be denoted by X.

Let $\overline{X}$ be the first task in the FFDL-schedule that FFDL can not fit when the Multifit schedule length is set at a time b, which is equal to or greater than 3/2 the optimal makespan or 3/2 the end of the last downtime. All tasks that are less than $\overline{X}$ are irrelevant to the fact that the FFDL schedule is unable to fit all tasks when the bound is set at time b, thus a minimal counterexample only contains tasks that are greater or equal to $\overline{X}$.

Lemma A.6 (Length of tasks)

A minimal counterexample contains only tasks that are greater or equal to the first task that can not be scheduled by FFDL Multifit within a time that is $\geq 3/2opt$.

In the following we normalize every time length to the length $\overline{X}$ of the task $\overline{X}$, that is, a number will represent that same number times the task length $\overline{X}$ in the measuring of time.

We continue by showing some more properties.

Lemma A.7 ($\geq$ 1 tasks in FFDL pretimes)

In a minimal counterexample the FFDL-schedule has at least one task in the pretime of each processor that has a pretime.

Proof: If a pretime on a processor p is empty in the FFDL-schedule, then the last task $\overline{X}$ did not fit in that pretime. But $\overline{X}$ is the least task, and thus the optimal schedule could also fit nothing in that pretime. Then we can build a lesser counterexample by maintaining the same processors and tasks with the difference that the mentioned pretime of p is replaced by downtime. Both the optimal schedule and the FFDL schedule will remain the same, and the new counterexample has fewer pretimes. $\triangle$

To better describe schedules on processors we will use the following notation for each processor schedule: [will denote start of the schedule, time 0, | will represent the downtime, and $[A_1A_2\ldots A_n|B_1B_2\ldots B_m|C_1C_2\ldots C_m$ will denote a schedule that has the tasks $A_1, A_2, \ldots A_n$ in the first pretime in the given order and the tasks $B_1, B_2, \ldots B_m$ in the given order in the second the pretime, and tasks $C_1C_2\ldots C_m$ after the pretimes.

We denote with $busy_{ALG}(p)$ length of the total processing time of processor p in an ALG-schedule, and $busy_{FFDL}(p)$ denotes the length of the busy time of FFDL on p when the Multifit bar is set at $max(3/2opt, max_{p\in p}(\gamma_p))$, and thus this schedule does not include $\overline{X}$ in a minimal counterexample.

Since the bound of 3/2 for Multifit FFDL in the case when there is at most one downtime on each machine was proved for uniform processors in Chapter III, a counterexample must contain at least one processor with two downtimes.

Lemma A.8 (Idle times in FFDL schedules)

The idle time in the FFDL-schedule of all pretimes and posttimes of a minimal counterexample, when the Multifit bar is set at $b \geq \frac{3}{2}max(opt, max_{p\in P}\gamma_p)$, is shorter than 1.

Proof: Suppose this is not the case and we have a processor p with a time slot ts which has an idle time that is greater or equal to 1. Then the task $\overline{X}$, which is assumed not to fit in any time slot when the algorithm attempts to schedule it with the Multifit bar set at b would fit in the time slot ts. $\triangle$

Lemma A.9 (Single tasks in time slots)

If in a pretime or a posttime ts, with $2 \leq ts < 3$ there is only one task $X_1 < 2$ in the optimal schedule and only one task $X_2 < 2$ in the FFDL-schedule of a minimal counterexample, we have $X_1 \leq X_2$.

Proof: Suppose $X_1 > X_2$. Let the tasks $X_{01} \geq X_{02} \geq .. \geq X_{0q}$ be all tasks that fulfill the inequality $X_1 \geq X_{0i} > X_2$. X_1 and all tasks X_{0i} are scheduled by FFDL in a pretime or posttime $ts_1 \leq ts$ and respectively $ts_{0i} \leq ts$, else they would have been scheduled in ts instead of task X_2. Reducing all tasks X_1 and X_{0i} to the length of task X_2 leaves the FFDL-schedule unchanged, since the order in which these are scheduled stays the same and no space ≥ 1 is created by reducing their length in this manner, while the optimal schedule can only get better. An schedule that is at least as good as the initial optimal schedule can be constructed by leaving the reduced tasks in the time slots in which they were scheduled by the optimal schedule on the initial problem instance.

This operation creates a lesser counterexample, thus the one initially considered, where $X_1 > X_2$, is not minimal.

Lemma A.10

In a minimal counterexample each pretime is of length 2 or longer than that.

Proof: Let $pre_i < 2$ be a pretime on a processor p. Then the optimal schedule can have at most one task X_1 in pre_i. If there is no task in the optimal schedule in pre_i,

pre_i can be filled with downtime and any tasks scheduled by FFDL in pre_i can be removed, creating a lesser counterexample, since the optimal schedule can only get better as it has less tasks to schedule, and the FFDL-schedule stays the same.

Suppose FFDL also scheduled a task X_2 in pre_i. Then by Lemma A.9 $X_2 \geq X_1$. Removing pre_i and X_2 we get a lesser counterexample, as the FFDL-schedule stays the same, while the optimal schedule can only get better, since replacing X_2 with the lesser task X_1 in the initial optimal schedule produces a schedule that at least as good as that schedule. Thus there is no pretime $pre_i < 2$ in a minimal counterexample. $\triangle$

As a consequence, the maximum between the end of the optimal schedule and the last end of a downtime is > 4 in a minimal counterexample, as we state in the following Lemma.

Lemma A.11 ($opt > 4$**)**
$max(C_{max}(OPT), max_{p \in P}(\gamma(p))) > 4$.

Lemma A.12 (Single tasks in FFDL schedules of time slots)
If the FFDL-schedule of two pretimes or posttimes $ts_1 \in [2, 3)$ and $ts_2 \in [2, 3)$ has exactly one task $X_1 < 2$ in ts_1 and exactly one task $X_2 < 2$ in ts_2, then $X_1 = X_2$.

Proof: Suppose $X_1 > X_2$. Then $ts_1 < ts_2$, as X_1 was scheduled by FFDL before X_2, and $X_1 < 2 \leq ts_2$ (also see Lemma A.10). Let X_{0i} with $X_2 < X_{0i} \leq X_1$ be all tasks scheduled by FFDL between after it scheduled X_1 and before it scheduled X_2. All these tasks have been scheduled alone in time slots $ts_i \leq ts_1$. Reducing X_1 and all tasks X_{0i} to the length of task X_2 creates a lesser counterexample, since the FFDL-schedule stays the same as no idle time that is ≥ 1 is created by these reductions, while the optimal schedule can stay the same or get better when the reduced tasks (and the other tasks) are left in their original positions (in the time slots where they were scheduled by the optimal schedule of the initial counterexample). △

Definition A.13 (Compensating processor)

A processor that has less busy time in the FFDL-schedule, without considering the last task $\overline{X}$, than in the optimal schedule is called a compensating processor.

Lemma A.14 (Existence of compensating processors)

There is at least one compensating processor.

Proof: The total busy time of the FFDL-schedule without the last task $\overline{X}$ is less than the total busy time of the optimal schedule since this one contains task $\overline{X}$. Thus there must be a processor on which OPT has more busy time than FFDL without task $\overline{X}$. △

B. General upper bound for FFDL Multifit

In this subsection we prove a general upper bound for the FFDL Multifit schedule length. We show that, given a problem instance with at most k downtimes on one processor, FFDL Multifit ends within $(\frac{3}{2} + \frac{1}{2k})opt$.

A problem instance the FFDL Multifit schedule of which ends after this bound also ends after time $\frac{3}{2}opt$, and so is a counterexample as defined above. Thus all theorems and lemmas from subsection A apply to it.

Lemma B.1 (Number of downtimes on *cp*)

A compensating processor *cp* belonging to a minimal counterexample has a number of downtimes that is equal to the maximum number of downtimes that appear on any processor of the counterexample.

Proof: We shall call k_p the number of downtimes on a processor p, and let k be the maximum number of downtimes on a processor in the given problem instance. By Lemma A.10 $opt \geq 2k$. Let $b \geq \frac{3}{2}opt$ be the schedule length set by Multifit when

FFDL failed to schedule $\overline{X}$ within the allocated time. A bound of $b+1$ would allow for scheduling all tasks, and so $C_{max}(Multifit) \leq b+1$.

Since *cp* is a compensating processor, and since in each pretime there is at most an idle time of 1, the maximum time difference between the end of the FFDL-schedule on *cp*, fdl_{cp} and the end of the optimal schedule on *cp*, opt_{cp}, is k_{cp}. Thus

$$opt + k_{cp} \geq opt_{cp} + k_{cp} > fdl_{cp} \geq C_{max}(Multifit) - 1 > \frac{3}{2}opt - 1 \geq opt + k - 1.$$

Thus $k_{cp} > k-1$, and so $k_{cp} = k$. △

Next we show the upper bound.

We have $C_{max}(Multifit) - 1 < opt_{cp} + k_{cp} \leq opt + k$. Then

$$\frac{C_{max}(Multifit)}{opt} < \frac{opt+k+1}{opt} = 1 + \frac{k+1}{opt} \underset{opt \geq 2k}{\leq} 1 + \frac{k+1}{2k} = 1.5 + \frac{1}{2k}.$$

For small k this bound is not very accurate, as we have shown in the previous chapter that FFDL Multifit finishes within $\frac{3}{2}opt$ if there is at most one downtime on each machine. In the next subsection we study the case when there are at most 2 downtimes on each machine. For $k \geq 3$, the above result implies that FFDL Multifit finishes within $\frac{5}{3}opt$, and the bound gets better as k increases.

C. Upper bound for k=2

In this subsection we consider the case when there are at most 2 downtimes on each machine.

Lemma C.1 (Optimal schedule length)

The length of the optimal schedule and the last end of a downtime are both < 6.

Proof: Suppose that is not the case. Then a compensating processor *cp* would have an FFDL-schedule busy time after the downtime of *cp* that is greater than

$\frac{1}{2}max(opt, max_{p\in P}\gamma_p) - 1 \geq 2$, since $post_{cp}$ has an idle time that is less than 1.

Also, the idle time in the pretimes of the FFDL-schedule of cp adds up to less than 2. Thus $busy_{FFDL}(cp) > pre_{1cp}-1+pre_{2cp}-1+busy_{opt.after}(cp)+2 \geq busy_{opt}(cp)$, and cp is not compensating. $\triangle$

Theorem C.2 (Structure of a compensating processor)

(a) a compensating processor cp has two downtimes,

(b) $busy_{opt}(p) \geq 2$ in each pretime.

(c) If the optimal schedule has at least a task after the downtime of cp then

(c1) $busy_{alg.after}(cp) > \frac{1}{2}opt$,

(c2) there is either one task $U_3 > 2.5$ and $U_3 > \frac{1}{2}opt$, or three tasks in the FFDL-schedule of the posttime of cp,

(c3) there is exactly one task U_i the FFDL-schedule of each pretime pre_{icp} of cp, and $U_1 + U_2 < 3$.

(d) If the optimal schedule has no task after the downtime of cp, when the Multifit bar is at 3/2 or higher and FFDL fails to schedule all tasks, the FFDL-schedule of cp is of type $[U_1|U_2|U_3$, where

(d1) $U_1, U_2, U_3 > \frac{1}{2}opt - 1$, and

(d2) All tasks U_1, U_2, U_3 are < 2, and $U_1 = U_2 = U_3$.

Proof: Suppose cp has no downtime. Then $busy_{alg}(cp) > \frac{3}{2}busy_{opt}(cp)$, and cp is not compensating. Suppose cp has only one downtime. Then the idle time in that downtime is < 1. We also have $busy_{alg.after}(cp) > \frac{1}{2}max(opt, \max_{p\in P}(\gamma(p))) - 1 \geq 1$, and so $busy_{alg}(cp) > busy_{opt}(cp)$. We have (a).

Suppose $busy_{opt}(ts) < 2$ in one of the pretimes ts of cp. Then at most one task is scheduled in the optimal pretime ts of cp, and this task is less long or equal to the task in the FFDL schedule of ts or there are more than one tasks in the FFDL-schedule of ts. In both cases $busy_{opt}(ts) \geq busy_{alg}(ts)$. Let $ts2$ be the other pretime of cp.

$busy_{opt}(ts2) - busy_{alg}(ts2) < 1$, and thus, since $busy_{alg.after}(cp) > busy_{opt.after}(cp) + 1$, we have $busy_{alg}(cp) > busy_{opt}(cp)$, and cp is not compensating. (b) follows.

Suppose the optimal schedule has a task X_3 after the downtime of cp. Then $busy_{alg.after}(cp) > X_3 + \frac{1}{2}opt - 1 \geq \frac{1}{2}opt$.

If there are two tasks scheduled by FFDL in any pretime of cp then the sum of the tasks scheduled by FFDL in the pretimes or cp is ≥ 3, and $busy_{FFDL}(cp) \geq \frac{1}{2}opt + 3 \geq \frac{1}{2}opt + \frac{1}{2}opt = opt$, and cp is not compensating. Thus each pretime pre_i of cp has exactly 1 task U_i scheduled in it.

If there are 2 tasks X_6 and X_7 scheduled by FFDL after the downtime in that order we have $X_6 + X_7 > \frac{1}{2}opt$, and $X_6 > \frac{1}{4}opt$ since it was scheduled first. Also if a pretime $pre_{icp} \geq post_{cp} > \frac{1}{2}opt + 1$, we must have $U_i > \frac{1}{2}opt$, and $U_i + busy_{alg.after}(cp) > opt$, and cp is not compensating. Thus $pre_{icp} < post_{cp}$, thus U_i was scheduled in pre_{icp} before X_6 in $post_{cp}$, and $U_i > \frac{1}{4}opt$. Then $U_1 + U_2 + busy_{alg.after}(cp) > \frac{1}{4}opt + \frac{1}{4}opt + \frac{1}{2}opt = opt$, and cp is not compensating. Thus the FFDL-schedule of the posttime of cp does not contain exactly two tasks.

There can not be more than 3 tasks in the FFDL-schedule of the posttime of cp, because then $busy_{FFDL}(cp) > 6 > opt$, and cp would not be compensating.

If there is only one task U_3 scheduled by FFDL after the downtime on cp, and $U_3 \leq pre_{icp}$ then $U_3 \leq U_i$, where U_i is the task scheduled by FFDL in pre_{icp}, in case $pre_{icp} \leq post_{cp}$. Then $U_3 + U_i \geq 2U_3 > opt$ If $pre_{icp} > post_{cp} = \frac{1}{2}opt + X_3$, then $U_i > pre_{icp} - 1 \ \geq \frac{1}{2}opt$, then $U_i + U_3 \geq opt$, and cp is not compensating.

Thus $U_3 > pre_{1cp}$ and $U_3 > pre_{2cp}$, and since $U_3 > \frac{1}{2}opt$, $U_3 > 2.5$ We have (c).

If the optimal schedule of a compensating processor does not have a task after the downtime, $busy_{alg.after}(cp) > \frac{1}{2}opt - 1$. Suppose there are two tasks or one task longer than or equal to 2 in one of the pretimes of cp. Then the other pretime can only have one task in it, else $busy_{alg}(cp) > 4 + \frac{1}{2}opt - 1 = 3 + \frac{1}{2}opt \geq opt$, and cp

would not be compensating.

Suppose there are 2 tasks in the FFDL-schedules of both pretimes of cp. Then $busy_{alg}(cp) > 4 + \frac{1}{2}opt - 1$, and cp would not be compensating. Suppose the FFDL-schedule has two tasks in one of the pretimes on cp. Let U_i be the task the FFDL-algorithm scheduled in the other pretime, pre_{icp}. Suppose there are two tasks in the FFDL-schedule of $post_{cp}$. Then the idle time in the pretimes of the FFDL-schedule, which must be greater than $busy_{alg.after}(cp)$ if cp is compensating, must be greater than 2, which is not possible, as each one must be less than 1. Thus $busy_{alg.after}(cp) < 2$, and there is only one task, U_3, in the posttime of cp.

Then $U_3 = U_i > \frac{1}{2}opt - 1$, from Lemma A.12 and $busy_{alg}(cp) = U_3 + U_1 + U_2 > opt$. Thus $busy_{alg}(pre_{icp}) < 2$, and there must be exactly one task, $U_i < 2$, in each pretime pre_{icp}, $i \in \{1, 2\}$. By Lemma A.12 and from the above deductions we have $U_1 = U_2 = U_3 > \frac{1}{2}opt - 1$, and $U_1 < 2$. $\triangle$

Theorem C.3 (No compensating processor of type C.2(c))

(a) If there is a compensating processor with a task after the downtime in the optimal schedule then there is a Y-task (a task the length of which is in the interval $[1.5, 2)$) in the FFDL-schedule of a time slot $ts \geq 2.5$.

(b) There is no compensating processor with the structure described in the statement (c) of Theorem C.2.

Proof: Suppose statement (a) is incorrect, and there is no Y-task in any time slot $ts \geq 2.5$. We use a weighing argument. Let X-tasks be in the interval $[1, 2)$ and have the weight 1, Z-tasks be in the interval $[2, \frac{1}{2}opt]$ and have weight 2, Z'-tasks be in the interval $(\frac{1}{2}opt, 3)$ and have the weight 3, R_i-tasks in the interval $[i, i + 1)$ and have the weight i for $i \in \{3, 4, 5\}$. There can be no tasks that are ≥ 6 by Lemma C.1.

Recall that the FFDL-schedule of any processor does not contain the task $\overline{X}$ when the multifit bar is set at a bound $b \geq 3/2max(opt, max_{p \in P}\gamma_p)$ whereas the optimal schedule does, and so the total weight of the tasks in the optimal schedule is greater than that of the FFDL-schedule.

We show that for every processor $p \in P$:

$$w_{alg}(p) \geq w_{opt}(p) \qquad (1),$$

where $w_{alg}(p)$ is the total weight of the FFDL-schedule of processor p, and $w_{opt}(p)$ is the total weight of its optimal schedule.

We also denote with $w_{alg}(ts)$, and $w_{opt}(ts)$ the total weight the FFDL-schedule and respectively that of the optimal schedule of time slot ts.

Let *cp* be a compensating processor as described in the statement (c) of Theorem C.2.

Let p be an arbitrary processor in a minimal counterexample. Then $opt \geq 5$, and for any processor p we have $post_p \geq \frac{1}{2}opt \geq 2.5$. According to the assumption at the beginning of this proof there are no tasks in the interval $[1.5, 2)$ in $post_p$. We know that $busy_{FFDL}(post_p) > post_p - 1 = 1.5$, and thus there are at least two X-tasks or a Z- or greater task scheduled by FFDL in $post_p$ and

$w_{alg}(post_p) \geq 2$ (a1).

Suppose p has no pretime.

If the weight of the optimal schedule of p is 0, 1, or 2, (1) holds, from $w_{alg}(post_p) \geq 2$.

Suppose $w_{opt}(post_p) = 3$. Then $busy_{opt}(p) \geq 3$, and $busy_{alg}(p) > post_p - 1 \geq 3 + \frac{1}{2}opt - 1 = 3 + 1.5 = 4.5$. Since there are no Y-tasks in the FFDL-schedule of p, at least 4 X-tasks, a Z-task and two X-tasks, an R_3-task and an X-task or an R_4- or greater task must be in the FFDL-schedule of $post_p$, and then $w_{alg}(p) \geq 4$.

Suppose $w_{opt}(p) = 4$. We either have an optimal schedule with $busy_{opt}(p) \geq 4$ or $busy_{opt}(p) > \frac{1}{2}opt + 1$. In the first case $busy_{alg}(p) > post_p - 1 \geq 4 + 1.5 = 5.5$. At least 4 X-tasks of length less than 1.5, a Z, Z', and R_3-task and 2 X-tasks, or a schedule containing an R_4 or greater task must be scheduled by FFDL in $post_p$, and $w_{alg}(p) \geq 4$.

If $busy_{opt}(p) \geq \frac{1}{2}opt + 1$, we have $busy_{alg}(p) > busy_{opt}(p) + \frac{1}{2}opt - 1 \geq opt \geq 5$. The argument above holds, except that this busy time can also be achieved by scheduling an R_3-task and one X-task in $post_p$, and $w_{alg}(p) \geq w_{opt}(p)$.

Suppose $w_{opt}(p) = 5$. Then $busy_{opt}(p) \geq 5$ or $busy_{opt}(p) \geq \frac{1}{2}opt + 2$. We have $busy_{alg}(p) > busy_{opt}(p) + \frac{1}{2}opt - 1 \geq 5 + 1.5 = 6.5$, or $busy_{alg}(p) \geq 4.5 + 1.5 = 6$. Since all X-tasks are less than 1.5, at least 5 X-tasks, a Z- or Z'-task and 3 X-tasks, an R_3-task and 2 X-tasks, an R_4-task and one X-task or an R_5-task must be scheduled by FFDL in $post_p$. We have $w_{alg}(p) \geq 5$.

The weight of the optimal schedule can not be 6 or greater, since $busy_{opt}(p) < 6$ by Lemma C.1, because 2 Z'-tasks scheduled on the same processor would have a length that is greater than that of the optimal schedule, and since one Z'-task Z_1 and additional busy time of 3 in the optimal schedule of p would imply that $opt \geq Z_1 + 3 > \frac{1}{2}opt + 3$, and then $\frac{1}{2}opt > 3$, contradicting again Lemma C.1.

Thus inequality (1) holds for all processors with no pretime.

Suppose p has one pretime. We call this pretime pre_p in the following argument. From Lemma A.10, $pre_p \geq 2$. Also, $w_{alg}(pre_p) \geq 1$, since the idle time in pre_p can't be less than 1. Suppose $w_{opt}(pre_p) \leq 2$.

Then, since $w_{alg}(post_p) \geq 2$, (1) holds when $w_{opt}(post_p) \in \{0, 1\}$.

Suppose $w_{opt}(post_p) = 2$. Then $busy_{alg}(post_p) > busy_{opt}(post_p) + \frac{1}{2}opt - 1 \geq 2 + 1.5 = 3.5$. Since the X-tasks in $post_p$ are less than 1.5, the FFDL-schedule of $post_p$ must contain at least 3 X-tasks, a Z or Z'-task and one X-task, or an R_3- or greater task in $post_p$, and $w_{alg}(post_p) \geq 3$. Then $w_{alg}(p) \geq 4 \geq w_{opt}(p)$.

Suppose $w_{opt}(post_p) = 3$. Then, as shown for the same situation ($w_{opt}(post_p) = 3$) when p was assumed to have no pretime, we have $w_{alg}(post_p) \geq 4$.

We can not have $w_{opt}(post_p) \geq 4$, since then $w_{opt}(p) \geq 6$, and we have shown above that this is not possible. The same argument as in the case when p has no pretime hold in this case as well, and in any situation in which the weight of the optimal schedule adds up to 6 or more than that. We shall thus not consider such cases in the remainder of this proof.

Suppose $w_{opt}(pre_p) = 3$. Then either $pre_p \geq busy_{opt}(p) \geq 3$, or $pre_p \geq busy_{opt}(p) > \frac{1}{2}opt$. In the first case we have $busy_{alg}(pre_p) \geq 2 = pre_p - 1$, and at least 2 X-tasks or a Z- or greater tasks must be in the FFDL-schedule of pre_p, and then $w_{alg}(pre_p) \geq 2$.

If a Z'-task is scheduled in the optimal schedule of pre_p, $busy_{alg}(pre_p) > \frac{1}{2}opt - 1 \geq 1.5$, since $opt > 5$. We also know that $pre_p > \frac{1}{2}opt \geq 2.5$, and so and X-task scheduled by FFDL in pre_p is < 1.5. Thus at least 2 X-tasks or a Z- or greater task are scheduled by FFDL in pre_p. Concluding, $w_{alg}(pre_p) \geq 2 \geq w_{opt}(pre_p) - 1$.

Note that for any weight $w_{opt}(post_p) \leq 2$ we have shown in the case when $w_{opt}(pre_p) = 2$ that $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$, using arguments that did not use length of the pretime, and so we also have $w_{alg}(p) \geq w_{opt}(p)$ for all processors where $w_{opt}(pre_p) = 3$.

Suppose $w_{opt}(pre_p) = 4$. If there is no Z'-task in the optimal schedule, we have $busy_{opt}(pre_p) \geq 4$, and $busy_{alg}(pre_p) \geq 3$. Since all X-tasks are less than 1.5, at least 3 X-tasks, a Z or Z'-task and an X-task, or an R_3 or greater task are scheduled by FFDL in the pretime of p, and $w_{alg}(p) \geq 3$. If there is a Z'-task in the pretime of p, we have $busy_{alg} > pre_p - 1 > busy_{opt}(pre_p) > \frac{1}{2}opt + 1 - 1 = \frac{1}{2}opt$ Suppose FFDL scheduled only two X-tasks X_1 and X_2 with $X_1 \geq X_2$ in pre_p.

Note that $pre_p > 3 > pre_{icp}$ for any $i \in \{1,2\}$. Thus $\frac{1}{4}opt < X_1 < U_i$. Then $busy_{alg}(cp) = U_1 + U_2 + busy_{alg.after}(cp) \geq \frac{1}{4}opt + \frac{1}{4}opt + \frac{1}{2}opt = opt$, and cp is not compensating. Thus at least 3 X-tasks, a Z- or Z'-task and an X-task or an R_3- or greater task are scheduled by FFDL in pre_p. Thus $w_{alg}(pre_p) \geq 3$. Since as shown above $w_{alg}(post_p) \geq 2$, $w_{alg}(p) \geq 5\ geq w_{opt}(p)$, since $w_{opt}(p) < 6$.

Suppose $w_{opt}(pre_p) = 5$. If there is no Z'-task in pre_p, we have $busy_{alg}(p) > pre_p - 1 \geq busy_{opt}(p) - 1 \geq 4$. If there is a Z'-task in pre_p, $busy_{opt}(pre_p) > \frac{1}{2}opt + 2$, and $busy_{alg}(pre_p) > busy_{opt}(pre_p) - 1 > \frac{1}{2}opt + 1 \geq 3.5$. At least 3 X-tasks (since $pre_p \geq 2.5$ and they must be less long than 1.5), a Z- or Z'-task and an X-task or an R_3- or greater task are scheduled by FFDL in pre_p, and $w_{alg}(p) \geq 3$. Since $w_{alg}(post_p) \geq 2$, we have $w_{alg}(p) \geq 5 \geq w_{opt}(p)$.

Suppose p has 2 pretimes pre_1 and pre_2. Without loss of generality we may assume that $w_{opt}(pre_1) \geq w_{opt}(pre_2)$.

We have $w_{alg}(pre_1) \geq 1$, and $w_{alg}(pre_2) \geq 1$.

Suppose $w_{opt}(pre_1) \leq 2$, and $w_{opt}(pre_2) \leq 2$. Then, if $w_{opt}(post_p) = 0$, (1) holds, since $w_{alg}(post_p) \geq 2$, and so $w_{alg}(p) \geq 4 \geq w_{opt}(p)$.

If $w_{opt}(post_p) = 1$, we have $busy_{alg}(post_p) > post_p - 1 \geq busy_{opt}(post_p) + \frac{1}{2}opt - 1 \geq \frac{1}{2}opt$. We have $post_p \geq \frac{1}{2}opt > pre_{icp}$, for $i \in \{1,2\}$. Thus if only two X-tasks X_1 and X_2, $X_1 \geq X_2$ are scheduled in $post_p$ we have $U_i \geq X_1 \geq \frac{1}{4}opt$, and $busy_{alg}(cp) = U_1 + U_2 + busy_{alg.after}(cp) > opt$, and cp is not compensating. Also note that $busy_{alg}(post_p) > Z_1$ for any Z-task Z_1. Thus at least 3 X-tasks, a Z-task and an X-task, or a Z'- or greater task are scheduled by FFDL in $post_p$. $w_{alg}(post_p) \geq 3$, and (1) holds.

Suppose $w_{opt}(pre_1) = 3$. Then by the same argument as in the case when p has only one pretime and $w_{opt}(pre_p) = 3$, we have $w_{alg}(pre_1) \geq 2$. Since $w_{alg}(post_p) \geq 2$ we have $w_{alg}(p) = w_{alg}(pre_1) + w_{alg}(pre_2) + w_{alg}(post_p) \geq 5 \geq w_{opt}(p)$. There was no loss of generality assuming $w_{opt}(pre_1) \geq w_{opt}(pre_2)$, so all possible cases for processors with two pretimes have been considered and (1) holds.

Thus statement (a) of this theorem holds. We proceed with proving statement (b).

Suppose there is a compensating processor *cp* as described in the statement (c) of Theorem C.2. Then, by statement (a) of this theorem there is at least one Y-task Y_1 in a time slot $ts \geq 2.5$. Let $i \in \{1, 2\}$. Suppose $pre_{icp} \geq 2.5$. Then $U_i \geq pre_{icp} - 1 \geq 1.5$. Suppose $pre_{icp} < 2.5$. Then U_i has been considered by FFDL before the Y-task was scheduled in ts, and thus $U_i \geq Y_1 \geq 1.5$. We have $busy_{alg}(cp) = U_1 + U_2 + busy_{alg.after}(cp) > 1.5 + 1.5 + \frac{1}{2}opt > \frac{1}{2}opt + \frac{1}{2}opt = opt$, and *cp* is not compensating. △

As a consequence there must be a compensating processor as described in statement (d) of Theorem C.2.

Lemma C.4 (U-task length)
Let $\epsilon = \frac{1}{2}opt - 2$. Let *cp* be a compensating processor, and U_1, U_2, and U_3, the tasks in its FFDL-schedule. Then

$$U_1 < 1.(3) + \frac{2}{3}\epsilon$$

and all U-tasks are less or equal to U_1 in a minimal counterexample.

Proof: Suppose this is not the case, and $U_1 \geq 1.(3) + 1.(6)\epsilon$. Since *cp* is compensating we have $busy_{opt}(cp) > 3U_1 \geq 4 + 2\epsilon = opt$, a contradiction.

Suppose there is a U-task $U_4 > U_1$. Then U_4 must have been scheduled by FFDL in a time slot $ts \leq pre_{1cp} < 3$, since otherwise it would have been scheduled in pre_{1cp}. Therefore U_4 is alone in its time slot, and by Lemma A.12 we have $U_4 = U_1$, contradiction. △

Theorem C.5 (Constraint for FFDL-schedule)

Let $\epsilon = \frac{1}{2}opt - 2$. The end of the optimal schedule or the maximum end of a downtime occur at a time that is greater or equal to 5, or the following statement is true:
(s1) there is a task $X \in (1 + \frac{1}{2}\epsilon, 1 + \epsilon]$ in a time slot $ts \in [3 + \epsilon, 3 + 2\epsilon)$.

Proof:

Let cp be a compensating processor, and U_1, U_2, and U_3 the tasks in its FFDL-schedule. We use the following task types and weights: X-tasks $\in [1, \frac{1}{2}opt - 1]$, with weight 1, U-tasks $\in (\frac{1}{2}opt - 1, 2)$ with weight $\frac{4}{3}$ if they are $< U_1$, and weight $\frac{4.(3)}{3}$ otherwise, Z_1-tasks $\in [2, \frac{1}{2}opt]$ with weight 2, Z_{21}-tasks $\in (\frac{1}{2}opt, \frac{1}{2}opt + \epsilon)$, with weight $2 + \frac{1}{3}$, Z_{22}-tasks $\in [\frac{1}{2}opt + \epsilon, 3)$, with weight $2 + \frac{2}{3}$. We shall call Z_2-task any task that is either a Z_{21}-task or a Z_{22}-task. In case $U_1 < 1.5$ we also consider a subcategory of Z_{22}-tasks, Z_{cp}-tasks $\in (2U_1, 3)$, with weight $\frac{8.6}{3}$. We will consider Z_{22}-tasks to be a subcategory of Z_2-tasks while knowing that all Z_2-tasks have a weight of $2 + \frac{1}{3}$ or greater. Furthermore we consider R_3-tasks $\in [3, 4)$ with subcategories R_{31}-tasks $\in [3, 3 + \epsilon)$ with weight 3 and R_{32}-tasks $\in [3 + \epsilon, min(3 + 2\epsilon, 4))$ and weight $3 + \frac{1}{3}$, and R_{33}-tasks $\in [3 + 2\epsilon, min(4, 3 + 3\epsilon))$ with weight $3 + \frac{2}{3}$, and R_{34}-tasks $\in [3 + 3\epsilon, 4)$ with weight 4. Last we consider R_4-tasks $\in [4, 5)$ with weight 4. A special type of U-task, U_{cp}, weighed at $\frac{4.(3)}{3}$, is made of tasks that equal to the U-tasks of the compensating processor, while all other U-tasks, which are weighed at 4/3, will be called U_a-tasks.

Define $w_{min}(ts)$ as the minimum weight that the FFDL-schedule of a minimal counterexample can have in a time slot of size ts when the Multifit bar is set at $b \geq \frac{3}{2}max(opt, max_{p \in P}\gamma_p)$ and FFDL fails to schedule all tasks. Also we denote with *opt* the maximum between the end of the optimal schedule and the last end of a downtime in this proof.

Let p^* be the processor on which FFDL schedules the last U-task. We next show that, assuming that $opt < 5$ and statement (s1) is not true, we have

$$\sum_{p \in P} w_{alg}(p) \geq \sum_{p \in P} w_{opt}(p) - 1/3,$$

which is untrue, since we know that

$$\sum_{p \in P} w_{alg}(p) = \sum_{p \in P} w_{opt}(p) - 1,$$

because the optimal schedule contains $\overline{X}$, while the FFDL-schedule does not.

For every processor p we show that:

(1) either p is not p^*, and $w_{alg}(p) \geq w_{opt}(p)$,

or $w_{alg}(p) \geq w_{opt}(p) - \frac{1}{3}$ when p is p^*.

Let p be an arbitrary processor. We shall first consider case (a), when processor p does not contain a U_{cp}-task in its schedule, except if the U_{cp}-task is in the posttime and p has exactly one pretime. Note that any U_{cp}-task is also a U-task, and is weighed higher than U-tasks.

Suppose p has no pretime. If $w_{opt}(p) \leq 1$, (1) holds, since the FFDL-schedule must have at least a task on p, as $post_p - 1 \geq \frac{1}{2}opt - 1 \geq 1$.

Suppose $w_{opt}(p) = 4/3$. Then there is a U-task in the optimal schedule after the downtime of p, and $busy_{opt}(p) > \frac{1}{2}opt - 1 = 1 + \epsilon$. Also, $\epsilon > 0$, since $opt > 4$.

Then $busy_{alg}(post_p) > \frac{1}{2}opt - 1 + \frac{1}{2}opt - 1 = opt - 2 \geq 2$. Since no single X- or U-task can fill this time, we have $w_{alg}(post_p) \geq 2$.

Suppose $w_{opt}(p) = 2$. Then $busy_{opt}(post_p) \geq 2$, and $busy_{alg}(post_p) > 2 + \frac{1}{2}opt - 1 \geq 3$. Note that $\frac{1}{2}opt - 1 < 5/2 - 1 = 1.5$, and so two X-task can not result in a busy time that is greater than 3, thus $w_{alg}(post_p) \geq 2 + \frac{1}{3}$.

Suppose $w_{opt}(p) = 2 + \frac{1}{3}$. Then $busy_{opt}(post_p) \geq \frac{1}{2}opt$, and $busy_{alg}(post_p) > \frac{1}{2}opt + \frac{1}{2}opt - 1 = opt - 1 > 3$. By the same argument as above, $w_{alg}(post_p) \geq 2 + \frac{1}{3}$.

Suppose $w_{opt}(p) = 2 + \frac{2}{3}$. Then there are 2 U-tasks in the optimal schedule. Then $busy_{alg}(p) > 2(\frac{1}{2}opt - 1) + \frac{1}{2}opt - 1 = \frac{3}{2}opt - 3 = 3 + 3\epsilon$. Two X-tasks or a U-task and an X-task would have a busy time that is less than $2 + 1 + \epsilon = 3 + \epsilon$, thus at least two U-tasks are needed to fill this time. Note that a Z_1 or a Z_2-task alone also can not fill $busy_{alg}(post_p)$. Then $w_{alg.after}(p) \geq 2 + \frac{2}{3}$.

Suppose $w_{opt}(p) = 3$. Then $busy_{alg}(post_p) > 3 + \frac{1}{2}opt - 1 \geq 4$. Two U-tasks are one Z_1 or Z_2-task are not enough to fill a time of 4, thus a combination of at least 3 tasks, a Z_1 or Z_2-task and an X or U-task, or an R_3- or greater tasks are scheduled in $post_p$. Thus $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(p) = 3 + \frac{1}{3}$. Then $busy_{opt}(p) > 2 + \frac{1}{2}opt - 1$, and $busy_{alg}(p) > 2 + 2(\frac{1}{2}opt - 1) = opt = 4 + 2\epsilon$. Three X-tasks add up to at most $3 + 3\epsilon < 3 + 2\epsilon + 1 < 4 + 2\epsilon$. At least 2 X-tasks and a U-task, or 2 U-tasks and an X-task, or 3 U-tasks, or a Z_1-task and two tasks or a Z_2- or greater task, or a Z_2-task and a U- or greater task, or an R_3-task and another task, or an R_4- or greater task are scheduled by FFDL in $post_p$, and $w_{alg}(post_p) \geq 3 + \frac{1}{3}$.

Suppose $w_{opt}(p) = 3 + \frac{2}{3}$. $busy_{opt}(p) > opt - 1$. $busy_{alg}(p) > opt - 1 + \frac{1}{2}opt - 1 = 3 + 2\epsilon + 1 + \epsilon = 4 + 3\epsilon$. A U-task and 2 X-tasks add up to less than $2 + 1 + \epsilon + 1 + \epsilon$ busy time, and thus $w_{alg}(post_p) \geq 3 + \frac{2}{3}$.

Suppose $w_{opt}(p) = 4$. Suppose there is no U-task in the optimal schedule. $busy_{alg}(post_p) > busy_{opt}(p) + \frac{1}{2}opt - 1 \geq 4 + 1 + \epsilon = 5 + \epsilon$. Two U-tasks and an X-task add up to less than $4 + 1 + \epsilon$, and so at least 3 U-tasks are needed to fill $post_p - 1 \geq 5 + \epsilon$. Also, at least Z_1- or Z_2-task and at least two tasks that are less than 2, or an R_3-task and a U- or greater task, or an R_4-task and an X- or greater task are needed to fill $post_p - 1$. Thus $w_{alg}(post_p) \geq 4$. Suppose there are U-tasks in the optimal schedule. Then the optimal schedule of p is made of 3 U-tasks. Then $busy_{alg}(post_p) > 4 + 4\epsilon$. We know any X-task is $< 1 + \epsilon$. Suppose 2 U-tasks U_4 and U_5, $U_4 \geq U_5$ can fill a time of $3 + 3\epsilon$. Then $U_1 \geq U_4 > 1.5 + 1.5\epsilon$, where U_1 is the task scheduled in pre_{1cp}., which can not happen by Lemma C.4. Thus at least 4 X-tasks, an X-task, 2 U-tasks and another task, or 3 U-tasks, or a Z_1-task or a Z_2-task and tasks of weight more than 2, a Z_{22}-task and a U-task or two X-tasks, an R_3-task and at least another task or an R_4- or greater task are scheduled by FFDL in $post_p$. We have $w_{alg}(post_p) \geq 4$.

Suppose $w_{opt}(p) = 4 + \frac{1}{3}$. Then $busy_{opt}(p) > \frac{1}{2}opt - 1 + 3 = \frac{1}{2}opt + 2$ $busy_{alg}(post_P) > 4 + \epsilon + 1 + \epsilon = 5 + 2\epsilon > 2 + 2 + \frac{1}{2}opt - 1 + \epsilon$. Suppose the FFDL-schedule of p is composed of 3 U-tasks. Then the average length of these tasks is $> (5 + 2\epsilon)/3 > 1 + 2/3 + 2\epsilon/3$. Since $post_p > post_{cp}$, the U-tasks of the compensating processors are all $> 1 + 2/3 + 2\epsilon/3$. Thus $busy_{opt}(cp) > busy_{alg}(cp) > 3(1 + 2/3 + 2\epsilon/3) = 3 + 2 + 2\epsilon > 5$, contradiction to the assumption that $opt < 5$.

Suppose $w_{opt}(p) = 4 + \frac{2}{3}$. Then $busy_{opt}(p) > 4 + 2\epsilon$. $busy_{opt}(p) > 2(\frac{1}{2}opt - 1) + 2 = opt$, contradiction.

Suppose p has one pretime pre_p. Here, we consider the case when there are no U_{cp}-tasks in $OPT(pre_p)$, but there can be U_{cp}-tasks in $OPT(post_p)$.

We know that $pre_p \geq 2$. Suppose $w_{opt}(p) \leq 2$. Suppose that there is a single task $U_0 < 2$ in the FFDL-schedule of pre_p. Then, by Lemma A.12 $U_0 = U_1 > \frac{1}{2}opt - 1$, where U_1 is the task in the first pretime of the compensating processor cp. Thus $w_{alg}(pre_p) \geq \frac{4.(3)}{3}$.

If $w_{opt}(post_p) = 0$, we have $w_{alg}(post_p) \geq \frac{4.(3)}{3}$, and thus $w_{alg}(p) > w_{opt}(p)$. If $w_{opt}(post_p) \in \{1, \frac{4}{3}, \frac{4.(3)}{3}\}$, we have $busy_{alg}(post_p) > 1 + \frac{1}{2}opt - 1 \geq 2$, and thus at least 2 tasks of length less than 2 or a Z_1- or greater task are scheduled in $post_p$. We have $w_{alg}(post_p) \geq 2$, $w_{alg}(p) \geq 3 + \frac{1.(3)}{3} \geq w_{opt}(p)$.

If $w_{opt}(post_p) = 2$. Then $busy_{alg}(post_p) > 2 + 1 + \epsilon$. A U- and an X-task can not fill this time ($U < 2$, $X \leq 1 + \epsilon$), thus at least 3 X-tasks, 2 U-tasks, or a Z_1- or Z_2-task and an X-task or an R_3- or greater task are scheduled by FFDL in $post_p$. Thus $w_{opt}(post_p) \geq 2 + \frac{2}{3}$, and $w_{alg}(p) \geq 4 \geq w_{opt}(p)$.

Suppose $w_{opt}(post_p) \in \{2 + \frac{1}{3}, \frac{2+1.(3)}{3}\}$. $busy_{alg}(post_p) > 2 + \epsilon + 1 + \epsilon = 3 + 2\epsilon$. Suppose the FFDL-schedule of $post_p$ is composed of 2 U-tasks. Then their average is $> 1.5 + \epsilon$, which is impossible by Lemma C.4. Thus 2 U-tasks can not be the FFDL-schedule of $post_p$. At least 3 X-tasks, a U-task and 2 X-tasks, a Z_1- or Z_2-task and an X-task, or an R_3- or greater task are scheduled by FFDL in $post_p$. Then $w_{alg}(post_p) \geq 3$. Note that we have just shown, since in the proof of this case we only used that $post_p \geq 4 + 2\epsilon$, that:

(C.5.2) $$w_{min}(4 + 2\epsilon) = 3.$$

$w_{alg}(p) \geq 4.(3)/3 + 3 = 2 + 2 + 1.(3)/3 \geq w_{opt}(p)$.

If $w_{opt}(post_p) \geq 2 + \frac{2}{3}$, we have $\gamma_p + busy_{opt}(post_p) > 2 + 2(\frac{1}{2}opt - 1) = opt$, a contradiction.

Suppose $w_{opt}(pre_p) = 2 + \frac{1}{3}$. Again $w_{alg}(pre_p) \geq \frac{4.(3)}{3}$. If $w_{opt}(post_p) = 0$, we have (1), and $w_{alg}(post_p) \geq \frac{4}{3}$.

Suppose $w_{opt}(post_p) = 1$ then $busy_{alg}(post_p) > 1 + 1 + \epsilon$, and $w_{alg}(post_p) \geq 2$, since no single task less than 2 can fill a busy time that is greater than 2.

Suppose $w_{opt}(post_p) \in \{\frac{4}{3}, \frac{4.(3)}{3}\}$. We have $busy_{alg}(post_p) > 1 + \epsilon + 1 + \epsilon$. No two X-tasks or Z_1-task alone can fill this time. At least 3 X-tasks, a U-task and an X-task, a Z_1-task and an X-task or a Z_2- or greater task are scheduled by FFDL in $post_p$. Thus $w_{alg}(post_p) \geq 2 + \frac{1}{3}$, and $w_{alg}(p) \geq 3 + \frac{2.(3)}{3} \geq w_{opt}(p)$.

Suppose $w_{opt}(post_p) = 2$. $busy_{alg}(post_p) > 2 + 1 + \epsilon$. Suppose 2 U-tasks could fill this time. Then their average is $> 1.5 + \frac{1}{2}\epsilon$, and since $pre_{1cp} < 3 < post_p$, we have $U_1 > 1.5 + \frac{1}{2}\epsilon$, and $busy_{alg}(cp) = 3U_1 > 4.5 + \frac{3}{2}\epsilon = \frac{1}{2}opt - 1 + 3.5 + \frac{1}{2}\epsilon =$

$\frac{1}{2}opt + 2.5 + \frac{1}{2}\epsilon \underset{opt<5}{>} opt + \frac{1}{2}\epsilon$, and cp is not compensating.

Suppose $w_{opt}(post_p) = 2 + \frac{1}{3}$. Then $busy_{opt}(p) = busy_{opt}(pre_p) + busy_{opt}(post_p) > \frac{1}{2}opt + \frac{1}{2}opt = opt$, contradiction. the same happens if $w_{opt}(post_p) > 2 + \frac{1}{3}$.

Suppose $w_{opt}(pre_p) = 2 + \frac{2}{3}$. Then $pre_p > 2 + 2\epsilon$. Suppose there is a single U-task U_0 in the pretime of p. Then we have $U_0 = U_1$ by Lemma A.12. If $w_{opt}(post_p) = 0$, (1) holds. If $w_{opt}(post_p) = 1$, $busy_{alg}(post_p) > 2 + \epsilon$. If $post_p > 3 + 2\epsilon$, $post_p - 1 > 2 + 2\epsilon$, and $w_{alg}(post_p) \geq 2 + 1/3$, since no two X-tasks and no Z_1-task can fill $busy_{alg}(post_p)$. Then (1) holds. If $post_p < 3 + 2\epsilon$, and only 2 X-tasks are in $OPT(post_p)$, at least one of them must be $> 1 + \frac{1}{2}\epsilon$, and we have statement (s1) of this theorem. Thus there are at least a U-task and an X-task, a Z_1-task and another task, a Z_2- or greater task, or three X-tasks in $FFDL(post_p)$. We have $w_{alg}(post_p) \geq 2 + 1/3$, and $w_{alg}(p) \geq w(U_1) + 2 + 1/3 \geq 2 + 2/3 + 1 = w_{opt}(p)$. Also we have shown that:

(C.5.3) $$w_{min}(3 + \epsilon) \geq 2 + \tfrac{1}{3}.$$

If $w_{opt}(post_p) \in \{4/3, 4.(3)/3\}$, $busy_{alg}(post_p) > 2 + 2\epsilon$, and $post_p \geq 3 + 2\epsilon$ since there must be is a U-task in $OPT(post_p)$. Then $busy_{alg}(post_p) > 2 + 2\epsilon$, and $post_p > 3 + 2\epsilon$. No two X-tasks or Z_1- or Z_{21}-task alone can fill $busy_{alg}(post_p)$. Suppose there are a U-task U_4 and an X-task in $post_p$. Suppose that after scheduling U_4 there is still a U-task U_5 that can be scheduled by FFDL. We have $U_5 \leq U_4$. Suppose $post_p - U_4 < U_5$. Then $U_4 \geq 1.5 + \epsilon$, and since we have $U_1 \leq U_4$, $busy_{opt}(cp) > 3U_1 \geq 4.5 + 3\epsilon \geq opt - 2 + 2.5 + 3\epsilon > opt$, contradiction. If $post_p - U_4 \geq U_5$, then we have two U-tasks in $post_p$, or at least a Z_{22}-task or a Z_{21} or a Z_1-task and another task, or three X-tasks in the FFDL-schedule of $post_p$, and $w_{alg}(post_p) \geq 2 + \frac{2}{3}$. Then (1) holds. The only other case is when there is no more U-task remaining to scheduled by FFDL after U_4, thus p is the processor with the last scheduled U-task by FFDL,

p^*, and we have $w_{opt}(p) \geq w_{opt}(pre_p) + 1 + \frac{4}{3} > 3 + \frac{2.(3)}{3} \geq w_{opt}(p) - \frac{1}{3}$. Note that this holds even when the U-task in $OPT(post_p)$ is a U_{cp}-task. We have that if p is not p^*

$$w_{min}(3 + 2\epsilon) \geq 2 + \tfrac{2}{3}, \tag{C.5.4}$$

and if p is p^*, then $w_{min}(3 + 2\epsilon) \geq 2 + \frac{1}{3}$.

If $w_{opt}(pre_p) = 3$ we have $w_{alg}(pre_p) \geq 2$, and by the same arguments as in the case before, which showed that in all cases that can occur for $post_p$ we have $w_{opt}(post_p) \leq w_{alg}(post_p) - 1$, (1) holds.

If $w_{opt}(pre_p) = 3 + \frac{1}{3}$, $pre_p \geq busy_{opt}(pre_p) = 3 + \epsilon$, and $busy_{alg}(pre_p) \geq 2 + \epsilon$. If we have 2 X-tasks in the FFDL-schedule of the pretime of p then there are X-tasks $> 1 + \frac{1}{2}\epsilon$ in time slots that are greater than $3 + \epsilon$, but less than $3 + 2\epsilon$ (else the time to fill would be at least $2 + 2\epsilon$, and two X-tasks add up to less than that), and we have statement (s1) of this theorem. Any Z_1-task is less than $2+2\epsilon$, so FFDL can not have scheduled only a Z_1-task in pre_p. If we have at least one task that is $\geq Z_2$ or two tasks one of which is a U-task, the weight difference to the optimal schedule's weight in pre_p is at most 1, and (1) holds.

Suppose $w_{opt}(pre_p) = 3+\frac{2}{3}$. $busy_{opt}(pre_p) > 3+2\epsilon$, and since $w_{min}(3+2\epsilon) \geq 2+\frac{2}{3}$, we have $w_{alg}(pre_p) \geq 2 + \frac{2}{3}$. Since $opt \leq 4 + 2\epsilon$, no task can be in $OPT(post_p)$. So (1) holds, since $w_{alg}(post_p) > 1$.

Suppose $w_{opt}(pre_p) = 4$: $OPT(pre_p)$ is either made of three U-tasks, or we have $busy_{opt}(pre_p) \geq 4$. No task can be fit after the downtime in the optimal schedule: if there were a task in the optimal schedule of $post_p$, in the first case we would have $busy_{opt}(p) > 3 + 3\epsilon + 1 = \frac{1}{2}opt + 2(\frac{1}{2}opt - 1) = opt + \frac{1}{2}opt - 2 > opt$, and in the second case the optimal schedule would be > 5.

Suppose $OPT(p)$ is made of 3 U-tasks. Then $pre_p > 3 + 3\epsilon > 3 + 2\epsilon$. Recall that $w_{min}(3 + 2\epsilon) = 2 + \frac{2}{3}$ by statement (C.5.4). Thus $w_{alg}(pre_p) \geq 2 + \frac{2}{3}$, and since $w_{alg}(post_p) > \frac{4}{3}$, and no task can be in $OPT(post_p)$, (1) holds.

Suppose $busy_{opt}(pre_p) \geq 4$. Then $busy_{alg}(pre_p) \geq 3$. Suppose there are 2 tasks only in the FFDL-schedule of pre_p. If one of them is ≥ 2 we have $w_{alg}(pre_p) \geq 3$, and since $w_{alg}(post_p) \geq 4.(3)/3$, (1) holds. Suppose both tasks, X_1 and X_2, with $X_1 \geq X_2$ are < 2. Since $pre_p > 3 > pre_{1cp}$, we have $X_1 \leq U_1$. Also $1.5 < (X_1 + X_2)/2 \leq X_1$. Since $opt < 5$ we have $\epsilon = opt/2 - 2 < 0.5$, and thus X_1 is a U-task. If X_2 is also a U- or greater task, we have $w_{alg}(p) \geq 2(4/3) + 4.(3)/3 > 4 = w_{opt}(p)$. Otherwise, since $post_p - X_1 \geq 2$, and any U-task can fit in that time, X_2-being an X-task implies that there are no more U-tasks after X_1 to schedule, and thus $p = p^*$. In this case we have $w_{alg}(p) \geq w_{opt}(p) - 1/3$.

Suppose $w_{opt}(pre_p) = 4 + \frac{1}{3}$. $busy_{opt}(pre_p) \geq 4 + 1 + \epsilon$. $busy_{alg}(pre_p) > 3 + \epsilon$. A U- and an X-task can not fill this space, since U tasks are less than 2 and X-tasks are $< 1 + \epsilon$. Thus at least 2 U-tasks, 3 X-tasks, a Z-task and another task, or an R_3- or greater task are scheduled in the FFDL-schedule of pre_p, and $w_{alg}(pre_p) \geq 3$ unless the FFDL-schedule of p is made of 2 U-tasks U_4 and U_5, in which case we have from Lemma C.4

$$pre_p - U_4 - U_5 \geq 4 + \epsilon - 2(1.(3) + 0.(6)\epsilon) = 1.(3) - 0.(3)\epsilon \underset{\epsilon<1}{>} 1.$$

Thus $FFDL(pre_p)$ can not be made of two U-tasks and we have $w_{alg}(pre_p) \geq 3$. Since $w_{alg}(post_p) \geq 4.(3)/3$, (1) holds. Also, we have shown that:

$$w_{min}(4 + \epsilon) \geq 3.$$

Suppose there are 2 pretimes on p. If both pretimes have weight 2 then, since in both pretimes and in the posttime there must be at least a U-task (or more than one task), (1) holds.

Suppose $w_{opt}(pre_{1p}) = 2 + 1/3$. Then $opt \geq pre_{1p} + pre_{2p} \geq 4 + \epsilon$. Suppose there is only one task U_5 in the FFDL-schedule on pre_{1p}, and only one task U_6 in pre_{2p}, and one task $U_7 < 2$ in $post_p$. Note that if there are more than one task or a task longer or equal to 2 in any of those time slots (1) holds. We have $U_5 = U_6 = U_7 = U_1$, and all these tasks are of type U_{cp}. (1) holds: If $U_1 < 1 + 2\epsilon$ we have $w_{alg}(p) = 4 + 1/3 = w_{opt}(p)$, and otherwise we have $w_{alg}(p) = 5$.

Next, we consider the case when there is at least one U_{cp}-task in the optimal schedule of p. Let all U-tasks that are not U_{cp}-tasks be called U_a-tasks.

Suppose there is no pretime on processor p. Suppose there is one U_{cp}-task on it. If the optimal schedule has only one U_{cp}-task on it, (1) holds, as $busy_{alg}(post_p) \geq U_1 + \frac{1}{2}opt - 1 \geq 2$, and thus $w_{opt}(p) \geq 2$.

If the optimal schedule has a U_{cp}-task U_{cp1} and an X-task or a U-task in $post_p$, we must have $busy_{alg}(p) > U_1 + 1 + \frac{1}{2}opt - 1 \geq U_1 + 2 + \epsilon$. Any X-task is $< \frac{1}{2}opt - 1$, any U-task that is not a U_{cp}-task is $< U_1$, and any U that was scheduled by FFDL in $post_p$ is $\leq U_1 \leq U_{cp1}$, since $post_p > pre_{1cp}$. Suppose there is a U-task or an X-task U_4 in $post_p$. We have $busy_{alg}(p) - U_4 > 2 + \epsilon$, a time that can be filled only with tasks that have a total weight of at least 2, and $w_{alg}(p) \geq 3 > w_{opt}(p)$. Suppose there is no U- or X-task in $FFDL(post_p)$. Any Z-task can not fill $busy_{alg}(p)$, and thus we have $w_{alg}(p) \geq 3 > 2w(U_1) \geq w_{opt}(p)$. We have also shown that

$$w_{min}(U_1 + 2\epsilon) \geq 3$$

Suppose there are 2 X-tasks X_1 and X_2 in the optimal schedule (in addition to the U_{cp}-task U_{cp1}). Three X-tasks are not enough to fill $busy_{alg}(cp) > 2 + U_1 + 1 + \epsilon \geq$

$2+1+\epsilon+1+\epsilon$, since $(X) < 2$, and $(X) < 1+\epsilon$ for any X-task (X). Thus if there are 3 X-tasks on p $w_{alg}(p) \geq 4$. If there are only 2 X-tasks X_4 and X_5 on p we have $busy_{alg}(p) - X_4 - X_5 > 2$, and $w_{alg}(p) \geq 4$.

Suppose there is at most 1 X-task on p. Suppose there is a U-task U_4 in the FFDL-schedule of p. We have $U_4 \leq U_1 \leq U_{cp1}$. $busy_{alg}(p) - U_4 \geq 3+\epsilon$. Any X- or U-task is < 2, and any X-task is $< 1+\epsilon$, thus at least 2 U- or greater tasks, a Z-task and at least an X-task, an R_3- or greater task, or 3 tasks are in the FFDL-schedule in addition to U_4 on p, and $w_{alg}(p) \geq 4/3 + 2 + 2/3 = 4 \geq 3 + \frac{2}{3} \geq w_{opt}(p)$.

Suppose there is no U-task on p. Suppose a Z_1- or Z_{21}-task Z_4 is on p. $Z_4 < 2+2\epsilon$. $busy_{alg}(p) - Z_4 > 2$, and $w_{alg}(p) \geq w(Z_4) + 2 \geq 4$. A Z_{22}-task and an X-task can only fill a time that is $< 3+1+\epsilon < busy_{alg}(p)$, and thus at least a Z_{22}-task and two tasks that are less than 2 or a U-task are needed to fill $busy_{alg}(p)$. We have in this case, again, $w_{alg}(p) \geq 4$. An R_3-task can not fill $busy_{alg}(p)$, and weighs at least 3, thus $w_{alg}(p) \geq 4 \geq w_{opt}(p)$ if such a task occurs in the FFDL-schedule of p. We have just shown that

$$w_{min}(U_1 + 3 + \epsilon) \geq 4$$

Suppose $OPT(p) = [|U_{cp1}U_{a1}X_1$, where the tasks are of types U_{cp}, U_a and X respectively. We have $w_{opt}(p) \leq 4$ and $busy_{alg}(p) > 3+2\epsilon+1+\epsilon$. Note that the busy time to be filled is greater than in the case above, and thus, as in the previous case we have $w_{alg}(p) \geq 4 \geq w_{opt}(p)$.

Suppose $OPT(p) \in \{[|U_{cp1}U_{a1}U_{a2}, [|U_{cp_1}(Z_4)\}$, where the first task is of type U_{cp} and the second and third tasks is of type U_a, and Z_4 is a Z_{22}-task. In case $U_{cp1} < 1+2\epsilon$ we have $busy_{alg}(p) > busy_{opt}(p) + 1 + \epsilon \geq U_{cp1} + 3 + 3\epsilon \geq 4 + 4\epsilon$. Suppose there are X-tasks in the FFDL-schedule of p. 4 X-tasks are not enough to fill this time, as each one is $< 1+\epsilon$, thus at least (a) 5 X-tasks, (b) 3 X-tasks and a U- or longer task, (c) 2 X-tasks or a Z_{21}-task (or Z_1-task) and a Z_{22} or greater task, or (d) 1 X-task and a U_a-task, a U_{cp}-task and tasks filling a busy time > 2, and thus which also have a weight ≥ 2, are in the FFDL-schedule of $post_p$, and we have $w_{alg}(p) \geq 4 + \frac{1}{3} \geq w_{opt}(p)$. Suppose there are no X-tasks in the FFDL-schedule of p.

Suppose there are no X-tasks in $FFDL(p)$. Any U_{cp}-task that is scheduled by FFDL in $post_p$ must be equal to U_1, since otherwise it would have been scheduled on the compensating processor. Thus any U-task on p is less or equal any U_{cp}-task. So, if there is one in FFDL(p) the remaining time to be filled is $> 3 + 3\epsilon$. If there is no second U-task on p we must have at least a Z-task and another task, or an R_3- or greater task, and $w_{alg}(p) \geq 4 + 1/3$. If there is a second U-task U_5, it is < 2, and so, any single task U_6 completing the FFDL-schedule of p must fulfill $U_5 + U_6 > 3 + 3\epsilon$, implying that, if U_6 is a U-task, $U_1 \geq max(U_4, U_5) \geq 1.5 + 1.5\epsilon$, contradiction to Lemma C.4. Thus more than one task < 2 or a task ≥ 2 completes the FFDL-schedule of p (together with the two U-tasks), and we have $w_{alg}(p) \geq 4 + 2/3 \geq w_{opt}(p)$. If there is no second U-task at least a Z-task and an X-task or an R_3-task are required to complete the FFDL-schedule of p, and $w_{alg}(p) \geq 4 + 1/3$.

Suppose there is no U-task or X-task on p. Two Z_1-tasks or two Z_{21} or a Z_1-task and a Z_{21}-task can not fill a time $> 4 + 4\epsilon$. Thus at least a Z_{22}-task and another Z-task, or an R_3 or R_4-task and another task greater than 2 are scheduled in $FFDL(post_p)$, and $w_{alg}(post_p) \geq 4 + 1/3$.

Suppose $OPT(p) = [|U_{cp1}U_{cp2}U_{a1}$. $busy_{alg}(p) \geq U_{cp1} + U_{cp2} + U_{a1} + 1 + \epsilon \geq 4 + 4\epsilon$. Since all X-tasks are less than $1 + \epsilon$, 4 X-tasks are not enough to fill this time. So if there are 4 X-tasks in the FFDL-schedule of p we have $w_{alg}(p) \geq 5$. If there are 3 X-tasks X_1, X_2, X_3 in the FFDL-schedule of p, we note that any U-task U_4 must be less than or equal to U_1, since $post_p$ is considered by FFDL after pre_{1cp}, and thus less than or equal to to U_{cp1} and U_{cp2}. So $busy_{alg}(cp) - X_1 - X_2 - X_3 - U_4 > 0$, and thus we have $w_{alg}(p) > 5 + 1/3$. If there is a task greater than a U-task on p in addition to the 3 X-tasks we have $w_{alg}(p) \geq 5$.

Suppose we have another U-task U_5, in addition to tasks X_1, X_2, U_4 instead of X_3 on p. Then we have $U_5 \leq U_{cp1}$, and $U_4 < U_{cp2}$, and $X_1 + X_2 < U_{a1} + 1 + \epsilon$, and so $X_1 + X_2 + U_4 + U_5 < busy_{alg}(p)$, and thus $w_{alg}(p) > 5 + 2/3$. Any combination of tasks that can fill $U_{a1} + 1 + \epsilon$ weighs more than 2, thus any possible FFDL-schedule on p if there are 2 U-tasks in it weighs at least $4/3 + 4/3 + 2 = 4 + 2/3 \geq w_{opt}(p)$.

Suppose there is only 1 U-task $U_4 \leq U_1 \leq U_{cp1}$ on p. The time remaining to be filled is $> U_{cp2} + U_{a1} + 1 + \epsilon$. 3 X-tasks are not enough to fill the remaining busy time. Any Z-task can also not fill a time that is > 3. An R_{31}-task is also not enough for that.Thus $w_{alg}(p) \geq w(U_4) + 3 + 1/3 \geq w_{opt}(p)$.

Suppose there are no U-tasks, and less than 3 X-tasks in the schedule of p. Suppose there are 2 X-tasks. The remaining busy time is $> U_{cp1} + U_{cp2}$. If this time can be filled by a Z-task it must be a Z_{22}-task and we have $w_{alg}(p) \geq 4 + 2/3 \geq w_{opt}(p)$. If a task greater than a Z-task is on p we have $w_{alg}(p) \geq 5$.

Suppose there at most one X-task on p. Suppose there is a Z_1- or a Z_{21}-task on p. The remaining busy time is $> 2 + 2\epsilon$. Any Z-task filling this time must be a Z_{22}-task, and we would have $w_{alg}(p) \geq 4 + 2/3$. If there is an X-task in this time, then there must be at least another task, that is neither an X- or a U-task, and we have $w_{alg}(p) \geq 5$. If there is an R_3- or greater task we have $w_{alg}(p) \geq 5 > w_{opt}(p)$.

Suppose there is a Z_{22}-task on p. The remaining busy time is > 2, and can be filled only with tasks the weight of which is at least 2. Then we have $w_{alg}(p) \geq 4{+}2/3$.

Suppose there is an R_{31}-task or R_{32}-task M_3 (and at most one X-task) in the FFDL-schedule of p. $busy_{alg}(p) - M_3 \geq 3 + 2\epsilon$, and since $w_{min}(3 + 2\epsilon) \geq 2 + 2/3$, we have $w_{alg}(p) \geq 3 + 2 + 2/3 > w_{opt}(p)$.

Suppose there is a R_{33}- or greater task M on p. If M is a R_{33}-task, $busy_{alg}(p) - M > 1 + \epsilon$, and so, since there are no U-tasks on p, $w_{alg}(p) > 3 + 2/3 + 2 > w_{opt}(p)$. Suppose M is an R_{34} or an R_4-task. We know that $M < 4 + 2\epsilon = opt < 4 + 4\epsilon < busy_{alg}(p)$. Thus $w_{alg}(p) \geq 5$.

Suppose $OPT(p) \in \{[|U_{cp1}X_1X_2X_3, [|U_{cp1}R_{31}, [|U_{cp1}Z_4X_1\}$, where Z_4 is a Z_1-task, U_{cp1} is a U_{cp}-task, and X_1, X_2, and X_3 are X-tasks.

We have $w_{opt}(p) = 3 + w(U_{cp1}) = 4.(3)/3$. We have $busy_{alg}(post_p) = busy_{alg}(p) > U_{cp1} + 3 + 1 + \epsilon > 5 + 2\epsilon$

Suppose the maximum task on p is an R_4-task. Then at least another task is needed to fill $busy_{alg}(p)$, and we have $w_{alg}(p) \geq 5$. Suppose the maximum task on p is an R_{32}-task. The remaining time to be filled is $> 1 + 2\epsilon$, and at least a U- or greater task or 2-tasks are needed for that. We then have $w_{alg}(p) \geq 3 + 1/3 + 4/3 = 4 + 2/3 \geq w_{opt}(p)$. Suppose the maximum task on p is an R_{31}-task. The remaining time to be filled is $> 2 + \epsilon$, and only tasks weighing at least 2 can do that. We have $w_{alg}(p) \geq 5$. Suppose the maximum task on p is a Z_{22}-task or a Z_{21}-task. The remaining time to be filled is $> 2 + 2\epsilon$. We have $w_{alg}(p) \geq 2 + 1/3 + 2 + 1/3$, since a Z_{21}-task or a U-task alone or 2 X-tasks are not enough to fill a time of $2 + 2\epsilon$.

Suppose the maximum task on p is a Z_1-task M_1. $busy_{alg}(p) - M_1 > 3 + \epsilon$. If there is another Z_1-task there must be at least another task and $busy_{alg}(p) \geq 5$. If there is a U-task U_4, we must have $U_4 \leq U_1 \leq U_{cp1}$, and $busy_{alg}(p) - U_4 - M_1 > 4 + \epsilon - M_1 > 2$, and $w_{alg}(p) \geq 2 + 4/3 + 2 = 5 + 1/3$.

Suppose the maximum task in $FFDL(p)$ is a U-task U_4. We have $U_4 \leq U_{cp1}$, and $busy_{alg}(p) - U_4 > 4 + \epsilon$. Suppose there is another U-task U_5. $busy_{alg}(p) - U_4 - U_5 > 2 + \epsilon$, a time length which can be filled only by tasks with a weight of at least 2. We have $w_{alg}(p) \geq 4/3 + 4/3 + 2 = 4 + 2/3 \geq w_{opt}(p)$.

Suppose there is no second U-task in $FFDL(p)$. There must be at least three tasks X_1, X_2, and X_3 in the FFDL-schedule of p in addition to U_4, since two X-tasks can not fill a time of 4. We have $1 + \epsilon > X_1$, and thus $X_2 + X_3 > 3$, and thus $max(X_1, X_2) > 1.5 \underset{opt<5}{>} \frac{1}{2}opt - 1$. Then there is a second U-task in pre_p, which contradicts the assumption at the beginning of this paragraph. Thus there must be at least 4 X-tasks in addition to U_4 on p. $w_{alg}(p) > 5$

Suppose there are only X-tasks on p. Since in the case above 3 X-tasks and a U-task were not enough to fill $busy_{alg}(p)$, neither are 4 X-tasks. Thus there must be at least 5 X-tasks, and $w_{alg}(p) \geq 5$.

Suppose $OPT(p) = U_{cp1}U_{cp2}U_{cp3}$. We have $busy_{alg}(p) > 3U_1 + 1 + \epsilon$. Suppose the maximum task on p is an R_4-task or an R_{34}-task. We know that $R_4 \leq 4 + 2\epsilon = opt$. Thus another task is needed to fill the time $busy_{alg}(p)$, and $w_{alg}(p) \geq 5$.

Suppose the maximum task on p, M_1, is an R_3-task that is less than $3 + 3\epsilon$, or a Z_{cp}-task. Then $busy_{alg}(p) - M_1 > U_{cp1}$, since $M_1 < 3 + 3\epsilon \leq U_{cp2} + U_{cp3} + 1 + \epsilon$. Thus at least a U-task $U_4 \leq U_{cp1}$ and another task or or 2 X-tasks, or a Z-task are necessary to fill this space. If a U-task U_4 is scheduled by FFDL in $post_p$ we must have $U_4 \leq U_1 \leq U_{cp1}$, because of U_4 was considered by FFDL after U_1, and because by definition U_1 has the least length a U_{cp}-task can have. So $w_{alg}(p) \geq 2 + 8.(6)/3 \geq w_{opt}(p)$.

Suppose the maximum task on p is a Z_2-task $M_2 < 2U_1$. If M_2 is a Z_{22}-task, we have $busy_{alg}(p) - M_2 > U_{cp1} + 1 + \epsilon \geq 2 + 2\epsilon$. At least a Z_2-task, or a U- and an X-task, or 3 X-tasks must also be in $FFDL(p)$, and $w_{alg}(p) \geq 5$. If M_2 is a Z_{21}-task or a Z_1-task. We have $busy_{alg}(p) - M_2 > U_{cp1} + U_{cp2}$, since $M_2 < 2 + 2\epsilon \leq 1 + \epsilon + U_{cp3}$. No other Z_{21}-task alone can fill this space, neither can two U-tasks that are $\leq U_1$, and thus, at least a Z_{21}-task and another task, or any combination of 3 tasks < 2 are also in the FFDL-schedule of p, and we have $w_{alg}(p) \geq 5 \geq w_{opt}(p)$.

Suppose the maximum task on p is a U-task U_4. Suppose there are two more U-tasks U_5 and U_6 in the FFDL-schedule of p. We have $U_4, U_5, U_6 \leq U_1 \leq U_{cpi}$ for $i \in \{1, 2, 3\}$. Thus at least another U-task or two X-tasks are needed to fill $busy_{alg}(p)$, and we have $w_{alg}(p) > 5$. Suppose there is only one more U-task U_5 in addition to U_4 on p. At least three X-tasks are needed to fill $busy_{alg}(p) - U_4 - U_5 > U_{cp3} + 1 + \epsilon$, and $w_{alg}(p) > 5$. Suppose there is no other U-task on p. At least 4 X-tasks are needed to fill $busy_{alg}(p) - U_4 > 3 + 3\epsilon$. Suppose there are only X-tasks on p. At least 5 X-tasks are needed to fill $busy_{alg}(p) > 4 + 4\epsilon$. We have $w_{alg}(p) \geq 5 \geq w_{opt}(p)$.

Suppose there is one pretime pre_p on p. Suppose there is a U_{cp}-task U_{cp1} in the optimal schedule of pre_p. If this is the only task in $OPT(pre_p)$, since there is at least a U_{cp}-task U_{cp2} or two tasks in $FFDL(pre_p)$, and $w_{alg}(post_p) > w_{opt}(post_p)$ as shown in the case when there was no U_{cp}-task in $OPT(pre_p)$, we have $w_{alg}(p) > w_{opt}(p)$.

Suppose there is also an X-task X_1 in $OPT(pre_p)$, and no other task. Any single task in $FFDL(pre_p)$ must be greater than $U_{cp1} \geq U_1$, and, if it is less than 2, equal to U_1, by Lemma A.12. Then we have $w_{alg}(pre_p) \geq 2$. If $w_{opt}(post_p) = 0$, we have $w_{alg}(post_p) \geq 4/3$, and (1) holds.

If $w_{opt}(post_p) \in \{1, 4/3, 4.(3)/3\}$ we have $busy_{alg}(post_p) \geq 1 + 1 + \epsilon$, and if there are only 2 X-tasks in $FFDL(post_p)$ we have statement (s1) of this theorem. $w_{alg}(post_p) \geq 2.(3)$. Then $w_{alg}(p) \geq 4.(3) > 3.(8) = 4.(3)/3 + 1 + 4.(3)/3 \geq w_{opt}(p)$.

Suppose $w_{opt}(post_p) = 2$. Then $busy_{alg}(post_p) > 2 + 1 + \epsilon$. We have shown above that in this case $w_{alg}(post_p) \geq 2 + 2/3$. Then $w_{alg}(p) \geq 2 + 2 + 2/3 \geq w_{opt}(p)$.

Suppose $OPT(pre_p) = U_{cp1}U_4$, where U_{cp1} is a U_{cp}-task and U_4 is another U-task. $busy_{alg}(pre_p) > U_1$, and thus $w_{alg}(pre_p) \geq 2$. Then $w_{opt}(post_p) < 2$, $w_{alg}(post_p) - w_{opt}(post_p) \geq 2.(3) - 4.(3)/3 = 2.(6)/3$ (from $w_{alg}(post_p) \geq 2.(3)$ when there is a single task < 2 in $OPT(post_p)$), and (1) holds.

Suppose $OPT(pre_p) = U_{cp1}X_1X_2$. If there are only 2 X-tasks in the FFDL-schedule of this pretime we have statement (s1) of this theorem. A Z_1-task is $< 2+\epsilon < busy_{opt}(pre_p)-1 < busy_{alg}(p)$. Thus $w_{alg}(pre_p) \geq 2+1/3$. If there is no task in the optimal schedule of $post_p$, (1) holds. If there is an X-task in $OPT(post_p)$, $busy_{alg}(post_p) > 1+1+\epsilon$, and if there are only 2 X-tasks in $FFDL(post_p)$ we have statement (s1) of this theorem. We have $w_{alg}(post_p) \geq 2+1/3$, since either a U- and an X-task or a Z_1-task and another task or at least a Z_2-task are scheduled in $FFDL(post_p)$. We have $w_{alg}(p) \geq 2+1/3+2+1/3 = 4+2/3 \geq w_{opt}(p)$.

Any task that is greater than an X-task can not be in $OPT(post_p)$, because then we would have $busy_{opt}(p) > 2(1/2opt-1)+2 = opt$, a contradiction.

Suppose $OPT(pre_p) = U_{cp1}U_4X_1$, where U_{cp1} is a U_{cp}-task, U_4 is a U-task, and X_1 is an X-task. $busy_{alg}(pre_p) \geq U_1+1+\epsilon$. If there is a R_3- or greater task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 3$. If there is only one Z-task in $FFDL(pre_p)$, it must be a Z_{cp}-task in case U_4 is a U_{cp}-task, and then $w_{alg}(pre_p) \geq 8.(6)/3$. Else there must be at least 2 U-tasks or 3 X-tasks in $FFDL(pre_p)$, and $w_{alg}(pre_p) \geq 3$. Concluding, if U_4 is a U_{cp}-task, $w_{alg}(p) \geq 8.(6)/3+4.(3)/3 > w_{opt}(p)$. Suppose U_4 is not a U_{cp}-task. Then, if there is only a Z-task in $FFDL(pre_p)$, it must be a Z_2-task, and $w_{alg}(pre_p) \geq 2.(3)$. Concluding, in all cases, $w_{alg}(p) \geq 2.(3)+4.(3)/3 = 3+2.(3)/3 = w_{opt}(p)$.

Suppose $OPT(pre_p) = U_{cp1}X_1X_2X_3$, where U_{cp1} is a U_{cp}-task and the other tasks are X-tasks. No task can be in $OPT(post_p)$. Also $w_{alg}(post_p) \geq U_1$, since if there is a single U-task in $post_p$ it will be equal to U_1 by Lemma A.12. We have $busy_{alg}(pre_p) \geq 3+\epsilon$. We have shown above that $w_{min}(4+\epsilon) \geq 3$, and thus $w_{alg}(pre_p) \geq 3$.

Suppose $OPT(pre_p) = U_{cp1}U_{cp2}XorU_1$, where $XorU_1$ is an X-task or a U-task of any type. No tasks fit in $OPT(post_p)$. We show $w_{alg}(pre_p) \geq \frac{8.(6)}{3}$. This holds when an R_3 or greater task or a Z_{cp}- or greater task are in pre_p. If the maximum task in pre_p is a Z-task M_2, which is not a Z_{cp}- task we have $pre_p - M_2 \geq 1$, and $w_{alg}(pre_p) \geq 3$. Suppose there are only tasks that are < 2 in pre_p. Any such task is less than the U_{cp}-tasks, thus at least three tasks that are less than 2 are scheduled in $FFDL(pre_p)$, or $\overline{X}$ would fit in pre_p in addition to its FFDL-schedule. Again $w_{alg}(p) \geq 3$. There are no tasks in $OPT(pre_p)$, but there must be at least a U_{cp}-task or a schedule of greater weight, and thus we have $w_{alg}(p) \geq \frac{8.(6)}{3} + \frac{4.(3)}{3} = 3(\frac{4.(3)}{3}) \geq w_{opt}(p)$.

Suppose there are two pretimes on p. Suppose without loss of generality that $pre_{1p} \leq pre_{2p}$. No U_{cp}-task fits in $post_p$ if $opt < 5$.

Suppose that a U_{cp}-task is in pre_{1p}. If there is no other task in $OPT(pre_{1p})$, (1) holds, as there are at least one U_{cp}-task or two tasks in each pretime and in $post_p$ in the FFDL-schedule. If $OPT(pre_{1p}) = U_{cp1} + X_1$, where X_1 is an X-task, we must have $w_{alg}(pre_1) \geq 2$, since any U-task U_4 scheduled in pre_{1p} by FFDL is $\leq U_1$, since $pre_{1p} \geq U_{cp1} + 1 \geq U_1 + 1 > pre_{1cp}$, and $U_4 < 2$ would have fit in pre_{1cp}.

No second U_{cp}-task can be scheduled on p, except if it is scheduled alone in pre_{2p}, in which case (1) holds. If it is scheduled in pre_{1p} we have $opt > pre_{2p} + 2 + 2\epsilon \geq opt$, and if it is scheduled with another task in pre_2 we have $opt \geq busy_{opt}(pre_{1p}) + busy_{opt}(pre_{2p}) > 2(1 + \epsilon + 1) \geq opt$.

Since pre_{1p} and pre_{2p} are exchangeable in this argument, we we conclude that (1) holds for p if there are two pretimes on p and there is a U_{cp}-task in $OPT(p)$. $\triangle$

The next definition is useful for writing more concise proofs. It assumes a given set of task types and a monotonic weight function defined for these task types.

Definition C.6 (Minimal configuration)

We call *minimal configuration* of a schedule of a time slot ts an ordered set of task types, ordered in decreasing order of their weights and sizes, such that a set of tasks of those types satisfy a certain condition such as a minimum busy time which results from the fact that the processor on which the time slot occurs is part of a minimal counterexample, such that the removal of any of the tasks from the set leads to the fact that the set does not satisfy the condition any more.

Lemma C.7

There are no tasks in the range $[2, 2+\epsilon]$ in time slots ≥ 3.

Proof: Any Z_1-task fits in $post_{cp}$, since $post_{cp} \geq 1/2opt = 2+\epsilon$. We know that a U-task, U_3, is scheduled in $post_{cp}$. If there is a Z_1-task M_1 in a time slot $ts \geq 2+\epsilon$, we have $M_1 \leq 2+\epsilon \leq post_{cp} < ts$, and $U_3 < M_1$, which implies that the FFDL would have scheduled M_1 in $post_{cp}$ and not in ts, since $post_{cp}$ was empty when M_1 was scheduled, and it would have been considered by FFDL before ts. $\triangle$

Next we show that statement (s1) of Theorem C.5 does not hold if $opt < 5$.

Theorem C.8 ((s1) does not hold)

Statement (s1) of Theorem C.5 does not hold if $opt < 5$.

Proof: We use the weights and notations from Theorem C.5 with the following exceptions: there are X_b-tasks in the range $(1+\epsilon/2, 1+\epsilon]$ weighed at 1.1(6), U-tasks $\in (1+\epsilon, 2)$ weighed at 1.(3) unless they are equal to U_1, with the subcategory U_{cp}-tasks of size U_1 which are weighed at 1.5, Z_2-tasks$\in (2+\epsilon, 3]$ are weighed at 2.(3), R_{32}-tasks$\in (3+\epsilon, 4]$ weighed at 3.(3), and R_{42}-tasks$\in \{4+\epsilon, 5\}$ weighed at 4.(3). We shall call X_a-tasks X-tasks that are not X_b-tasks, Z_1-tasks Z-tasks that are not Z_2-tasks, and R_{31}-tasks R_3-tasks that are not R_{32}-tasks.

Let p^{**} be the processor on which the last X_b-task is scheduled by FFDL. We show that, unless $p = p^{**}$, $w_{alg}(p) \geq w_{opt}(p)$, and otherwise $w_{alg}(p) \geq w_{opt}(p) - 0.5$.

Statement (s1) implies that:

(a3) there are no tasks in the range $(1+\epsilon, 3+\epsilon]$ in time slots which are greater than $3+2\epsilon$.

Suppose there is no pretime on p.

Suppose $w_{opt}(post_p) = 0$. Then $w_{alg}(post_p) \geq 1.5$.

Suppose $w_{opt}(post_p) \in \{1, 1.1(6)\}$. Then $busy_{alg}(post_p) \geq 2+\epsilon$. If there are only 2 X-tasks in this time slot then at least one of them is a X_b-task, and either $p = p^{**}$ and $w_{alg}(post_p) \geq 2.1(6)$ or $w_{alg}(post_p) \geq 2.(3)$. If there is a U- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 2.(3)$.

Suppose $w_{opt}(post_p) \in \{1.(3), 1.5, 2\}$. $busy_{alg}(post_p) > 2 + 2\epsilon$, and $post_p > 3 + 2\epsilon$. Two X-tasks can not fill this time. Suppose there is a U-task or a Z-task in $FFDL(post_p)$. This can not be by statement (a3). Thus $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) \in \{2.1(6), 2.(3), 2.5, 2.(6), 2.8(3), 3\}$. In all possible cases $busy_{opt}(post_p) \geq 2 + \frac{1}{2}\epsilon$. $busy_{alg}(post_p) > 3(1 + \frac{1}{2}\epsilon)$. Thus, if there are only 3 X-tasks in $FFDL(post_p)$, at least one of them is a X_b-task, and we have: if $p = p^{**}$ $w_{alg}(post_p) \geq 3.1(6)$, and else $w_{alg}(post_p) \geq 3.5$, if there is enough place for 3 X_b-tasks in $post_p$. Suppose there is not enough place. If a third X_b-task does not fit in $post_p$ the other two must add up to $4+3/2\epsilon-1+\epsilon = 3+1/2\epsilon$, which is not possible for 2 X-tasks if $opt < 5$.

Suppose $w_{opt}(post_p) \in \{3.1(6), 3.(3), 3.5\}$. $busy_{alg}(post_p) > 3 + \epsilon/2 + 1 + \epsilon$. Suppose 3 X-tasks can fill this time. Then one of them has to be $> 4/3 + \epsilon/2$, and so it is a X_b-task. The other two tasks need to add up to $3+\epsilon/2$, so at least a second one must be a X_b-task and be $> 1.5 + \epsilon/4$. However, since $opt < 5$, $\epsilon < 0.5$, and there can not be any X-tasks that are > 1.5. Thus 3 X-tasks can not fill this time, and if there are only X-tasks in $FFDL(post_p)$, $w_{alg}(post_p) \geq 4$. If there is a R_{32} or greater task in $FFDL(post_p)$, we have $w_{alg}(post_p) \geq 3.5$.

Suppose there is a U_{cp}-task, a U-task and an X-task in $OPT(post_p)$, and we have $w_{opt}(post_p) \in \{3.8(3), 4, 4.08(3), 4.1(6), 4.25\}$ or $busy_{alg}(post_p) > U_1+1+\epsilon+1+1+\epsilon \geq 4+3\epsilon$ If there is a R_{32}- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 4.(3)$. If there are only X-tasks in $FFDL(post_p)$ there must be at least 4 of them, since 3 of them can not add up to $U_1 + 2 + (1+\epsilon) + \epsilon$. If there are more than 4 tasks $w_{alg}(post_p) \geq 5$. Since $busy_{alg}(post_p) > 4 + 2\epsilon$, at least one of the four tasks is a X_b-task. Thus, either $p = p^{**}$, and $w_{alg}(post_p) \geq 4.1(6)$, or $p \neq p^{**}$, and $w_{alg}(post_p) \geq 4.5$, as any three X_b-tasks fit in a time of length $(1+\epsilon) + (1+\epsilon) + 2$.

Suppose $w_{opt}(post_p) \in \{4, 4.1(6), 4.25, 4.(3), 4.5\}$, and there is no U_{cp}-task in $OPT(post_p)$ except when the weight is 4.5. $busy_{alg}(post_p) > 4 + 1 + \epsilon$. If there is a R_3- $(+1+\epsilon)$ or greater task in $FFDL(post_p)$ we have $w_{alg}(post_p) \geq 5$. Else we have X-tasks. 3 or less of them add up to $< 2 + 2 + 1 + \epsilon$. If there are 4 of them, one of them has a length $> 1 + 0.25 + \epsilon/4$. We know that $\epsilon < 1$, and thus $0.25 > eps/4$. Then at least one of the tasks is a X_b-task. Thus, if $p = p^{**}$, $w_{alg}(post_p) \geq 4.1(6)$, and else $w_{alg}(post_p) \geq 4.5$.

Suppose there are 3 U_{cp}-tasks in $OPT(post_p)$. Then $busy_{alg}(post_p) > 4(1+\epsilon)$. If there is a R_4- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 4.5$. If there is a R_{33}-task $w_{alg}(post_p) \geq 5$, and the same happens if we have a smaller R_3-task. If there are only X-tasks there must be at least 5 of them, since all X-tasks are $< 1 + \epsilon$, and $w_{alg}(post_p) \geq 5$. In all cases $w_{alg}(post_p) > w_{opt}(p)$

Suppose there is a pretime on p. Then there can not be 2 U-tasks in $OPT(post_p)$, else $pre_p + busy_{opt}(post_p) > opt$.

Suppose $w_{opt}(pre_p) \leq 2$. Then $w_{alg}(pre_p) \geq 1.5$. For $w_{opt}(post_p) \leq 3.25$ we have $w_{alg}(post_p) \geq w_{opt}(pre_p) + 0.5$, unless $p = p^{**}$ or $p = p^*$, and else $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.1(6)$.

Suppose $w_{opt}(pre_p) \in \{2.1(6), 2.(3)\}$. Then $w_{opt}(post_p) < 3$. $w_{alg}(pre_p) \geq 1.5$, and the statement to prove holds, since in these cases $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.8(3)$, or $p = p^{**}$ and $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.5$.

Suppose $w_{opt}(pre_p) \in \{2.5, 2.(6), 3\}$. Then, $w_{alg}(pre_p) \geq 2$ since $busy_{alg}(pre_p) > pre_p - 1 \geq busy_{opt}(pre_p) - 1 \geq U_1$. Also, $w_{opt}(post_p) < 2.(3)$, since $busy_{opt}(post_p) < 2 + \epsilon = \frac{1}{2}opt \leq opt - busy_{opt}(pre_p)$. In these cases we have $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$, and the statement to prove holds.

Suppose $w_{opt}(pre_p) \in \{3.1(6), 3.(3)\}$. There are no Z_1-tasks in $FFDL(pre_p)$. If there is a Z_2-task, we have $w_{alg}(pre_p) \geq 2.(3)$, and the same happens if there is a U-task in FFDL(pre_p). If $pre_p < 3 + \epsilon$, there are two X_b-tasks in pre_p. Else there must be at least one X_b-task in pre_p, and, if $p = p^{**}$, $w_{alg}(pre_p) \geq 2.1(6)$, and else $w_{alg}(pre_p) \geq 2.(3)$. Then, unless FFDL scheduled the last X_b-task in pre_p, $w_{alg}(pre_p) \geq w_{opt}(pre_p) - 1$, and else $w_{alg}(pre_p) \geq w_{opt}(pre_p) - 1.1(6)$, and the statement to prove holds.

Suppose $w_{opt}(pre_p) \in \{3.5, 3.(6), 3.8(3)\}$. $busy_{alg}(pre_p) > U_1 + 1$. $pre_p > 3 + \epsilon$. If there is a U-task U_4 in $FFDL(pre_p)$, there is also a X-task in $OPT(pre_p)$. If there is a R_3-task, we have $w_{alg}(pre_p) \geq 3$. Else there must be either three or more X-tasks in pre_p, or at least one of the two tasks is a X_b-task. Then, if $p = p^{**}$, $w_{alg}(pre_p) \geq 2.1(6)$, and else $w_{alg}(pre_p) \geq 2.(3)$. In either case, the statement to prove holds, as $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.5$, as no task fits in the optimal schedule of $post_p$.

Suppose there are 2 U_{cp}-tasks in pre_p. Then $w_{opt}(pre_p) \geq 3$. $busy_{opt}(pre_p) \geq U_1 + 1 + \epsilon$. No single task < 2 can fill $pre_p - 1$, and be scheduled alone in pre_p, since then, by Lemma A.12 they would be equal. Thus $w_{alg}(pre_p) \geq 2$, and the statement to prove holds.

Suppose the optimal configuration on pre_p is $U_{cp}U_{cp}X > 3 + 2\epsilon$. $busy_{alg}(post_p) > 2 + 2\epsilon$, and at least 3 X-tasks are needed to fill this time. $w_{alg}(pre_p) \geq 3 = w_{opt}(pre_p) - 1$, and (1) holds.

Suppose $w_{opt}(pre_p) \in \{4, 4.1(6), 4.(3), 4.5\}$. $busy_{alg}(pre_p) > 3$. If $opt < 5$, 3 X-tasks are needed to fill this time, and (1) holds.

Suppose $w_{opt}(pre_p) \in \{4.(6), 4.8(3)\}$. These weights represent configurations of the optimal schedule for which $busy_{opt}(p) > 4 + 2\epsilon > opt$, so they can not occur.

Suppose there are 2 pretimes on p. If $opt < 5$ there can not be 3 tasks in either pretime. We assume $w_{opt}(pre_{1p}) \geq w_{opt}(pre_{2p})$ for convenience. If $w_{opt}(pre_{1p}) > 2.(6)$, there must be 2 U-tasks or tasks of length at least 3 in pre_{1p}, which can not occur.

Suppose $w_{opt}(pre_{1p}) = 2.(6)$. Then there are a U_{cp}- and a X_b-task in pre_{1p}. Then $busy_{alg}(pre_{1p} > U_1$, $w_{alg}(pre_{1p}) \geq 2.(3)$, or $pre_{1p} > 3 + \epsilon$ (which can not occur if $opt < 5$). We have $w_{opt}(pre_{2p}) \leq 2$, and $w_{opt}(p) \leq 4.(6) < 2.(3) + 1.5 + 1.5 = 5.(3) \leq w_{alg}(p)$.

Suppose $w_{opt}(pre_{1p}) = 2.5$. Then there are a U-task and a X_b-task in $FFDL(pre_{1p}$. We have $w_{opt}(pre_{2p}) \leq 2$, and $w_{opt}(p) \leq 4.5 \leq 3w(U_1) \leq w_{alg}(p)$.

Suppose $w_{opt}(pre_{1p}) = 2.(3)$. Then there are a U-task and a X-task or 2 X_b-tasks in $FFDL(pre_{1p}$. We have $w_{opt}(pre_{2p}) \leq 2.1(6)$, and $w_{opt}(p) \leq 4.5 \leq w_{alg}(p)$.

Suppose $w_{opt}(pre_{1p}) \leq 2.1(6)$. We have $w_{opt}(pre_{2p}) \leq 2.1(6)$, and $w_{opt}(p) \leq 4.5 \leq w_{alg}(p)$. △

Theorem C.9 (Constraint if $opt \geq 5$**)**

If $opt \geq 5$, one of the following statements is true:

(s2) There is a task $X \in (1 + \epsilon/2, 1 + \epsilon]$ in a time slot $ts \in (3 + \epsilon, 3 + 2\epsilon)$. X is not the only task of length $\leq 1 + \epsilon$ in $FFDL(ts)$.

(s3) There is a task $X_1 \in (1.(3), 1 + \epsilon]$ in the FFDL-schedule of a time slot $ts \geq 5$. There are at least two other tasks the length of which is less than or equal to $1 + \epsilon$ in ts.

Proof: We consider the following types of tasks: X-tasks $\in [1, \frac{1}{2}opt - 1]$, with weight 1, U-tasks $\in (\frac{1}{2}opt - 1, 2)$ with weight 1.5, Z-tasks $\in [2, 3)$ with subcategories Z_1-tasks$\in [2, \frac{1}{2}opt]$ with weight 2 and Z_2-tasks $\in (\frac{1}{2}opt, 3)$ with weight 2.5, and R_i-tasks $\in [i, i+1)$ with weight i, for $i \in \{3, 4, 5\}$. No task of length 6 or greater can exist, since $opt < 6$.

Suppose statements (s2) and (s3) do not hold. We show that for every $p \in P$,

(1) $$w_{alg}(p) \leq w_{opt}(p),$$

which is a contradiction, since the FFDL-schedule does not contain the task $\overline{X}$, while the optimal schedule does, and $\sum_{p \in P} w_{alg}(p) = \sum_{p \in P} w_{opt}(p) - 1$.

Let $p \in P$ be an arbitrary processor. Let cp be a compensating processor, and U_1, U_2, and U_3 be the tasks in the FFDL-schedule of pre_{1cp}, pre_{2cp}, and $post_{cp}$. Let w_{min} be defined as in the proof of Theorem C.5.

Let $\epsilon = \frac{1}{2}opt - 2$. Note that $\epsilon \geq 0.5$, since $opt \geq 5$, and that $\frac{1}{2}opt - 1 \geq 1.5$. Thus all U-tasks are ≥ 1.5.

Suppose p has one pretime pre_p.

We know that there must be at least a U- or greater task or two tasks in $FFDL(pre_p)$, and if $pre_p < 3$ and there is only one task in $FFDL(pre_p)$, it must be equal to U_1 by Lemma A.12. So if $pre_p < 3$ we have $w_{alg}(pre_p) \geq 1.5$.

Suppose $w_{opt}(pre_p) \leq 2$. $w_{alg}(pre_p) \geq 1.5$. Suppose there is no task in $OPT(post_p)$. Then there must be at least a U-task or two tasks in $FFDL(post_p)$ and we have $w_{alg}(post_p) \geq 1.5$.

Suppose there is an X-task in $OPT(post_p)$, and $w_{opt}(post_p) = 1$. $busy_{alg}(post_p) > post_p - 1 \geq 1 + 1.5 = 2.5$. No single task of length less than 2 can fill this time, and so $w_{alg}(post_p) \geq 2$.

Suppose $w_{opt}(post_p) = 1.5$. $busy_{alg}(post_p) > 1 + \epsilon + 1 + \epsilon$. No two X-tasks can fill this time, and we have $w_{alg}(post_p) \geq 2.5$.

Suppose $w_{opt}(post_p) = 2$. Then $busy_{alg}(post_p) > 2 + 1 + \epsilon$, and no U-task (< 2) and X-task ($< 1 + \epsilon$) or two X-tasks, or Z-task alone can fill this time, and we have $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) = 2.5$. Then $busy_{alg}(post_p) > 1 + 1 + \epsilon + 1 + \epsilon$. As in the previous case $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) \geq 3$. No two U-tasks can be in $OPT(post_p)$, since then we would have $opt > 2 + 2 + 2\epsilon \geq opt$, contradiction. Also $busy_{opt}(post_p) < 4$, else $opt \geq pre_p + busy_{opt}(post_p) \geq 6$. Thus $busy_{opt}(post_p) \geq 3$, $post_p \geq 3 + 2 + \epsilon$, and $busy_{alg}(post_p) \geq 3 + 1 + \epsilon$. A Z-task (< 3) and an X-task ($< 1 + \epsilon$) can not fill this time. If 3 X-tasks fill this time, then at least 1 X-task which is > 1.5 is in $FFDL(post_p)$, and we have statement (s3). Else $w_{alg}(post_p) \geq 3.5$.

Concluding (1) holds in all cases when $w_{alg}(pre_p) \leq 2$.

Also,

$$\text{if } w_{opt}(post_p) \leq 2 \text{ we have } w_{alg}(post_p) \geq w_{opt}(post_p) + 1. \tag{2}$$

Suppose $w_{opt}(pre_p) = 2.5$. We have $busy_{opt}(post_p) < 3$, else $busy_{opt}(p) \geq \frac{1}{2}opt + 3 > opt$, and $w_{opt}(post_p) \neq 2.5$, else $busy_{opt}(p) > 2(2 + \epsilon) = opt$.

So $w_{opt}(post_p) \leq 2$, and (1) holds, by statement (2) which we have shown above.

Suppose $w_{opt}(pre_p) = 3$. $busy_{alg}(p) > 2$, and we have $w_{alg}(p) \geq 2$, since no single task that is < 2 can fill this time. Again $w_{opt}(post_p) \leq 2$, and we have from statement (2) from above: $w_{opt}(p) \leq w_{alg}(pre_p) + 1 + w_{alg}(post_p) - 1 = w_{alg}(p)$.

Suppose $w_{opt}(pre_p) = 3.5$. If there are 2 X-tasks in this time slot we have statement (s2) of this theorem. Else we have $w_{alg}(pre_p) \geq 2.5$, and (1) holds.

Suppose $w_{opt}(pre_p) = 4$. Then $busy_{opt}(pre_p) \geq 4$, since any combination of tasks resulting in this weight will fulfill this statement. $busy_{alg}(pre_p) \geq 3$. If $FFDL(pre_p)$

is made of 2 X-tasks $X_1 \geq X_2$ we must have $busy_{alg}(post_p) \leq 2 + 2\epsilon$, and $post_p < 3 + 2\epsilon$. Also we know that $(X_1 + X_2)/2 \geq 1.5 > 1 + \epsilon/2$, and $4 > 3 + \epsilon$. Thus we have (s2).

Else there is a U-task in $FFDL(pre_p)$, and we have $w_{alg}(pre_p) \geq 2.5$. No Z-task can fill $busy_{alg}(pre_p)$, and thus we have $w_{alg}(pre_p) \geq 2.5$ in all cases. If $w_{opt}(post_p) = 0$, we have $w_{alg}(post_p) \geq 1.5$, and (1) holds. If $w_{opt}(post_p) = 1$, we have $post_p \geq 3 + \epsilon$, and $busy_{alg}(post_p) > 2 + \epsilon$. Only a Z_2- or greater task can fill this time by itself. If $post_p \geq 3 + 2\epsilon$, and there are two tasks with length < 2 in $FFDL(post_p)$, at least one of them must be a U-task, and we have $w_{alg}(post_p) \geq 2.5$, and (1). Else, if there are only 2 X-tasks in $FFDL(post_p)$, we have statement (s2). Thus, in all cases we have either statement (s2) or $w_{alg}(post_p) \geq 2.5$, and (1). No U- or greater task can be in $OPT(post_p)$, since then $busy_{opt}(p) > 1/2opt - 1 + 4 > opt$.

Suppose $w_{opt}(pre_p) = 4.5$. $busy_{alg}(pre_p) > 3 + \epsilon$. One U- or smaller task (< 2) and one X-task ($< 1 + \epsilon$), or one Z-task, can not fill this time. Thus $w_{alg}(pre_p) \geq 3$. Since no task can be in $OPT(post_p)$, else $busy_{opt}(p) \geq 4 + \epsilon + 1 = \frac{1}{2}opt + 3 > opt$, we have (1).

Suppose $w_{opt}(pre_p) = 5$. We have $w_{opt}(post_p) = 0$, and $w_{alg}(post_p) \geq 1.5$. $busy_{alg}(pre_p) > 4$.

Suppose the maximum task in pre_p is a Z_2-task. Then at least another task is scheduled in $FFDL(pre_p)$, and we have $w_{alg}(pre_p) \geq w_{opt}(pre_p) - 1.5$, and (1) holds. If the maximum task in $FFDL(pre_p)$ is an R_3-task, at least another task is scheduled there, and we have $w_{alg}(pre_p) \geq 4$, and (1). The same happens if an R_4-task is scheduled in $FFDL(pre_p)$.

Suppose there is a Z_1-task M_1 in pre_p. If $post_p \geq pre_p \geq 5$, then $w_{alg}(post_p) > 3$, and (1) holds. Else, since $2 + \epsilon \leq post_p < pre_p$, there must be another Z_1-task $M_2 \geq M_1$ in pre_p, else M_1 would have been scheduled in $post_p$. Also, $M_1 < pre_p - 2$, so there must be at least one other task in $FFDL(pre_p)$ in addition to M_1. So we have $w_{alg}(p) \geq 5 = w_{opt}(p)$.

Suppose the maximum task in $FFDL(pre_p)$ is a U-task U_4. We have $busy_{alg}(pre_p) - U_4 > 2$, and so $w_{alg}(pre_p) \geq 1.5 + 2 = 3.5$, and then $w_{alg}(p) \geq w_{opt}(p)$. If there are only X-tasks in $FFDL(pre_p)$ we have the case when there are at least 4 of them and (1) holds, and the case when there are only 3 of them and we have statement (s3), as at least one must be $> 4/3$ for the three of them to fill $busy_{alg}(pre_p) > 4$.

We can not have $w_{opt}(pre_p) \geq 5.5$, since if $w_{opt}(p) = 5.5$, we have $busy_{opt}(p) > 5 + \epsilon \geq \frac{1}{2}opt + 3 > opt$, a contradiction, and $w_{opt}(p) < 6$, since $opt < 6$, and all weights in the current proof are less or equal to the lengths of the tasks to which they are assigned.

Suppose there is no pretime on p. We have shown above that $w_{alg}(post_p) > w_{opt}(post_p)$ when $w_{opt}(post_p) \leq 2.5$.

Suppose $w_{opt}(post_p) = 3$. Then $busy_{alg}(post_p) \geq 4 + \epsilon$. We have shown above that $w_{min}(3 + \epsilon) \geq 3$, when we considered the case $w_{opt}(post_p) = 2$, so (1) holds in this case too.

Suppose $w_{opt}(post_p) = 3.5$. Then $busy_{alg}(post_p) \geq 4 + 2\epsilon$. If there is an R_4-task in $FFDL(post_p)$, (1) holds. If there is an R_3-task, then at least one other task is needed to fill $busy_{alg}(post_p)$. If there is a Z-task M_2 in $FFDL(post_p)$ we have $busy_{alg}(post_p) > 1 + 2\epsilon \geq 2$, and the remaining weight of FFDL in $post_p$ at least 2, thus $w_{alg}(post_p) \geq 4 > 3.5 = w_{opt}(p)$. Suppose there are only U- and X-tasks in $FFDL(post_p)$. Suppose there is at least a U-task U_4. We have $busy_{alg}(post_p) - U_4 > 2 + 2\epsilon$, and since $w_{min}(2 + 2\epsilon) \geq 2.5$, as 2 X-tasks or a Z_1-task can not fill this time, we have $w_{alg}(post_p) \geq 4 > w_{opt}(post_p)$. If there is no U-task in $FFDL(post_p)$, we must have at least 3 X-tasks, as two X-tasks can't fill a time of 4. From Lemma C.4 we have $U_1 < 1.(3) + \frac{2}{3}\epsilon < \frac{1}{3}busy_{alg}(post_p)$, and any X-task is less long than U_1.

Thus there are at least 4 X-tasks in $FFDL(post_p)$, and we have $w_{alg}(post_p) \geq 4$. Note that in all cases we had $w_{alg}(post_p) \geq 4$, and so $w_{min}(4+2\epsilon) \geq 4$.

Suppose $w_{opt}(post_p) = 4$. $busy_{alg}(p) > 5+\epsilon > 4+2\epsilon$. Then $w_{alg}(post_p) \geq w_{min}(4+2\epsilon) \geq 4 \geq w_{opt}(p)$.

Suppose $w_{opt}(post_p) = 4.5$. $busy_{alg}(post_p) > 3+1+\epsilon+1+\epsilon = 5+2\epsilon$. If there is an R_4-task in $FFDL(p)$, there is at least another task there and $w_{alg}(p) \geq 5$. If there is an R_3-task M_3 in $FFDL(p)$, we have $busy_{alg}(post_p) - M_3 > 1+2\epsilon \geq 2$, and $w_{alg}(p) \geq 5$. Suppose there is a Z_2-task M_2 in $FFDL(post_p)$. Then $busy_{alg}(post_p) - M_2 > 2+2\epsilon$, and $w_{alg}(p) \geq 2.5+1.5 = 5$, since no two X-tasks or Z_1-task can fill this time. Suppose there is a Z_1-task M_1 in $FFDL(post_p)$. Then $busy_{alg}(post_p) - M_1 > 3+\epsilon$. Note that a time of $3+\epsilon$ can only be filled with tasks the total weight of which is at least 3: a U- or X-task must be < 2, and an X-task is $< 1+\epsilon$, and a Z-task can not fill this time. Thus $w_{alg}(p) \geq 5$ Suppose the maximum task in $FFDL(p)$ is a U-task U_4. $busy_{alg}(post_p) - U_4 > 3+2\epsilon > 3+\epsilon$. We have just shown that a time of $3+\epsilon$ can only be filled by tasks the total weight of which is at least 3, and thus $w_{alg}(p) > 1.5+3$. Suppose there are only X-tasks in $FFDL(post_p)$. Let $X_1 \geq X_2 \geq X_3$ be the longest X-tasks in $FFDL(p)$. We have $busy_{alg}(p) - X_1 > 4+\epsilon$, $busy_{alg}(p) - X_1 - X_2 > 3$. If X_3 and only another X-task are in $FFDL(p)$ we have $X_3 > 3/2 = 1.5$, and we have statement (s3) of this theorem. Else there must be two more tasks in $FFDL(p)$ and we have $w_{alg}(p) \geq 5$.

Suppose $w_{opt}(p) \geq 5$. We can not have $w_{opt}(p) \geq 5.5$, as shown above. Thus $w_{opt}(p) = 5$. We have $busy_{alg}(p) > 6+\epsilon \underset{\epsilon<1}{>} 5+2\epsilon$. We have shown in the previous case that when $busy_{alg}(post_p) > 5+2\epsilon$, and there is either an R_4-task, an R_3-task, or a Z_2- or Z_1-task on p we have $w_{alg}(p) \geq 5$. Suppose there is a U-task U_4 on p. Then $busy_{alg}(post_p) - U_4 > 4+\epsilon \geq 4.5$. If three X-tasks fill this time we have statement (s3) of this theorem, and else we have $w_{alg}(p) \geq 1.5+3.5 = 5$. Suppose there are only X-tasks in $FFDL(p)$. If there are only 4 of them we have statement (s3), and else we have $w_{alg}(p) \geq 5$.

Suppose there are 2 pretimes on p. Suppose $w_{opt}(post_p) = 0$. Note that $w_{alg}(p) \geq 4.5$, since the minimum possible total weight of the FFDL-schedule is achieved when all time slots of p have a U-task in them. If $w_{opt}(pre_{ip}) \leq 2.5$ and $w_{opt}(pre_{jp} \leq 2$ we have (1), where $i \neq j$, and $w_{opt}(pre_{ip}) \geq w_{opt}(pre_{jp})$.

If both pretimes have a weight that is ≥ 2.5, we have $busy_{opt}(p) > 1/2opt + 1/2opt = opt$, contradiction. If $w_{opt}(pre_{ip}) \geq 3$, then $w_{alg}(pre_{ip}) \geq 2$, and $w_{alg}(p) \geq 5 = w_{opt}(p)$.

Suppose there is a task in $OPT(post_p)$. If it is a U- or greater task, or if there is more than one task, we have $opt > 4 + \frac{1}{2}opt - 1 = 1/2opt + 3 > opt$. Thus there is a single X-task in $OPT(post_p)$. Both pretimes must have a weight that is ≤ 2, else we have $opt > 2 + w_{opt}(pre_{ip}) + 1 \geq 3 + \frac{1}{2}opt > opt$. Also, $w_{alg}(post_p) \geq 2$, since $busy_{opt}(post_p) \geq 2 + \epsilon$. So $w_{alg}(p) \geq w_{opt}(p)$.

Theorem C.10 (Tasks $\geq$ 1.5)

Let $\epsilon = \frac{1}{2}opt - 2$. If $opt \geq 5.(3)$ there are no tasks ≥ 1.5 and $\leq 1 + \epsilon$ in a time slot that is ≥ 4.

Proof: Suppose $opt \geq 5.(3)$ and there are tasks ≥ 1.5 in a time slot $ts_1 \geq 4$.

Consider the following tasks types: X-tasks which are either X_a-tasks $\in [1, 1.(3))$, weighed 1 or X_b-tasks $\in [1.(3), 1.5)$ weighed 1.1(6) Y-tasks $\in [1.5, U_1)$ weighed 1.5, U_{cp}-tasks $= U_1$, weighed 1.(6) if $U_1 < 1.8(3)$ and 1.(7) otherwise, Z-tasks which are either Z_1-tasks $\in [2, 2 + \epsilon]$ weighed 2 or Z_2-tasks $\in [2 + \epsilon, 3)$ weighed 2.5, R_i-tasks $\in [i, i + 1)$ weighed i for $i \in \{3, 4, 5\}$.

Let p^* be the processor on which $FFDL$ scheduled the last Y-task and p^{**} be the processor on which it scheduled the last X_b-task. We show that (1): if $p \neq p^*$

and $p \neq p^{**}$, $w_{alg}(p) \geq w_{opt}(p)$, and else, if $p = p^{*}$, $w_{alg}(p) \geq w_{opt}(p) - 0.5$, and else, if $p = p^{**}$, $w_{alg}(p) \geq w_{opt}(p) - 0.(3)$.

We next consider $post_p$. If $w_{opt}(post_p) = 0$, $w_{alg}(post_p) \geq 1.(6)$.

Suppose $w_{opt}(post_p) \in \{1, 1.1(6), 1.5, 1.(6), 1.(7), 2\}$ $busy_{alg}(post_p) \geq 1 + 1 + \epsilon$. If $w_{opt}(post_p) = 1$, we have $w_{alg}(post_p) \geq 2.5$.
If $w_{opt}(post_p) \geq 1.1(6)$, $busy_{alg}(post_p) \geq 1.(3) + 1 + \epsilon \geq 3$. If there is a Z_2- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 3$. Else, if there is a Y-task in $FFDL(post_p)$, either $p = p^{*}$, and $w_{alg}(post_p) \geq 2.5$, and else $w_{alg}(post_p) \geq 3$. If there is no Y-task in $FFDL(post_p)$, there must be at least three X-tasks and again $w_{alg}(post_p) \geq 3$, unless $p = p^{*}$.

Suppose $w_{opt}(post_p) \in \{2.1(6), 2.(3), 2.5, 2.(6), 3, 3.(3)\}$. $busy_{alg}(post_p) > 2.(3) + 1 + \epsilon \geq 4$. If there is a Z_2- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 3.5$. Else, if there is a Y-task Y_1 in $FFDL(post_p)$, $busy_{alg}(post_p) - Y_1 > 2$, and $w_{alg}(post_p) \geq 3.5$. Else, if there are only X-tasks, if there are only 3 of them and we don't have $p = p^{**}$, $w_{alg}(post_p) \geq 3.5$. Else $w_{alg}(post_p) \geq 3.1(6)$.

Note that in the cases $w_{opt}(post_p) \in \{2.(7), 2.8(3), 2.9(4), 3.1(6), 3.2(7)\}$, we have $busy_{alg}(post_p) > busy_{opt}(post_p) + 1 + \epsilon > 4.5 = 2.8(3) + 1 + 0.(6)$. Recall that a U_{cp}-task weighing 1.(7) has a length $> 1.8(3)$. The relevant minimal configurations for $FFDL(post_p)$ are R_5, R_4, R_3X, Z_2Y, YYX, YX_aX_a, $XXXX$, and $w_{alg}(post_p) \geq 4$ unless $p = p^{*}$ and the last Y-task is scheduled by FFDL in $post_p$, in which case $w_{alg}(post_p) \geq 3.5$.

Suppose $w_{opt}(post_p) \in \{3.5, 3.(5), 3.(6), 3.(7), 3.9(4), 4\}$. $busy_{alg}(post_p) > 3.5 + 1 + \epsilon$. If there is R_3- or greater task in $FFDL(post_p)$ $w_{alg}(post_p) \geq 4$. Else if there is a Z_2-task M_2 in $FFDL(post_p)$, $busy_{alg}(post_p) - M_2 > 1.5 + \epsilon$, and $w_{alg}(post_p) \geq 4.5$. If there is a Y-task Y_1 in $FFDL(post_p)$, $busy_{alg}(post_p) - Y_1 > 2.5 + \epsilon \geq 3$. If there is a second Y-task, $w_{alg}(post_p) \geq 4$, and else $w_{alg}(post_p) \geq 4.5$, since 2 X-tasks can not fill a time of 3. We have $w_{alg}(post_p) \geq 4$.

Suppose $w_{opt}(post_p) \in \{4.(1), 4.1(6), 4.2(7), 4.(3), 4.(4), 4.5, 4.(5), 4.(6), 4.7(2), 5\}$. $busy_{alg}(post_p) > 4.(3) + 1 + \epsilon \geq 6$. If there is a R_3- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 5$. If there is a Z_2-task M in $FFDL(post_p)$, $busy_{alg}(post_p) - M \geq 3$, and $w_{alg}(post_p) \geq 5$. Suppose there is a Y-task Y_1 in $FFDL(post_p)$ If there is a second Y-task, $w_{alg}(post_p) \geq 5$. Else, since $busy_{alg}(post_p) - Y_1 > 4$, at least 3 X-tasks are in $FFDL(post_p)$, and at least one of them is $\geq 1.(3)$. Then $w_{alg}(post_p) \geq 4.(6)$ if $p = p^{**}$, and $w_{alg}(post_p) \geq 5$ otherwise.

Suppose $w_{opt}(post_p) \in \{5.0(5), 5.1(6), 5.(3), 5.(4), 5.5\}$. $busy_{alg}(post_p) \geq 5.(3) + 1 + \epsilon \geq 7$. If there is a R_3- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 5.5$. Else if there is a Z_2- task M in $FFDL(post_p)$, $busy_{alg}(post_p) - M > 4$, and thus another Z_2-task is not enough to fill this time. If there is a Y-task Y_1 in the remaining time, $busy_{alg}(post_p) - M - Y_1 > 2$, and $w_{alg}(post_p) \geq 2.5 + 1.5 + 2 = 6$. If there are only X-tasks in the remaining time, at least 3 are necessary to fill a time > 4, and $w_{alg}(post_p) \geq 5.5$. If the maximum task in $post_p$ is a Y-task, the remaining time to fill is > 5. If there is a second Y-task, the remaining time to fill is > 3, and $w_{alg}(post_p) \geq 5.5$. If there are only X-tasks in $FFDL(post_p)$, there must be at least 5 of them, and their average is $> 7/5 > 1.(3)$. Then $w_{alg}(post_p) \geq 5.1(6)$ if $p = p^{**}$, and $w_{alg}(post_p) \geq 5.5$ otherwise.

We have just shown that on processors without pretimes the statement to prove holds. Suppose there is a pretime on p.

Suppose $w_{opt}(pre_p) \leq 2$. Then $w_{alg}(pre_p) \geq 1.(6)$. We have $w_{opt}(post_p) \leq 3.5$, and there can not be any 2 U-tasks in $FFDL(post_p)$. Suppose $w_{opt}(post_p) = 3.(3)$, and there are 2 X_b-tasks in $OPT(post_p)$. $busy_{alg}(post_p) \geq 3.(6) + 1 + \epsilon \geq 5.(3)$. The same argument as in the case when $w_{opt}(post_p) = 3.5$ holds, and $w_{alg}(post_p) \geq 4$. (1) holds.

Suppose $w_{opt}(pre_p) = 2.1(6)$. $w_{alg}(pre_p) \geq 1.(6)$. $w_{opt}(post_p) \leq 3.5$, $w_{opt}(p) \neq 3.(3)$, else $opt \geq 6$, and the statement to prove holds.

Suppose $w_{opt}(pre_p) = 2.(3)$. $w_{alg}(pre_p) \geq 1.(6)$. $w_{opt}(post_p) \leq 3$, and the statement to prove holds, unless $w_{opt}(post_p) = 3$. In that case $busy_{alg}(post_p) \geq 4 + \epsilon$, and $\epsilon \geq 1.(6)/2 = 0.8(3)$. If there is a R_3- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 4$. If there is a Z_2-task in $FFDL(post_p)$, the remaining time filled by the FFDL-schedule is $> 1.8(3)$, and $w_{alg}(post_p) \geq 4$. If there is a Y-task in $FFDL(post_p)$, the remaining time to fill is 2.8(3), and at least 2 X_b-tasks are necessary to fill this time, and thus $w_{alg}(post_p) \geq 3.8(3)$ in case $p = p^*$, and $w_{alg}(post_p) \geq 4$ otherwise. If there are only X-tasks in $FFDL(post_p)$, there must be at least 4 of them, and $w_{alg}(post_p) \geq 4$. In all cases the statement to prove holds.

Suppose $w_{opt}(pre_p) = 2.5$. $w_{alg}(pre_p) \geq 1.(6)$. $w_{opt}(post_p) \leq 3.1(6)$, and the statement to prove holds unless $w_{opt}(pre_p) = 3$. In that case $busy_{alg}(post_p) \geq 4 + \epsilon$, and $\epsilon \geq 5.5/2 - 2 = 0.75$. The relevant minimal configurations are R_4, R_3X, Z_2Y, YX_bX, $YX_aX_aX_a$, $XXXX$, and $w_{alg}(post_p) = 4$ or $p = p^*$ and $w_{alg}(post_p) = 3.(6)$. The statement to prove holds in this case.

Suppose $w_{opt}(pre_p) \in \{2.(6), 2.(7), 2.9(4)\}$. The optimal schedule of pre_p is either a U_{cp}-task and an X-task or a Y-task and an X_b-task. In the first case we have $w_{alg}(pre_p) \geq 2$, since $busy_{alg}(pre_p) > U_1$, and any single task < 2 in pre_p would be $\leq U_1$ by Lemma A.12, and then (1) holds. In the second case $w_{opt}(pre_p) = 2.(6)$, $busy_{opt}(pre_p) \geq 2.8(3)$, thus $busy_{opt}(post_p) < 3.1(6)$. Suppose $U_1 > 1.8(3)$. Then $w_{alg}(pre_p) \geq 1.(7)$. Again $w_{opt}(post_p) \leq 3$, and the statement to prove holds in all cases except $w_{opt}(post_p) = 2.(6)$. Suppose $w_{opt}(post_p) = 2.(6)$. $busy_{alg}(post_p) > 2.8(3) + 1 + \epsilon \geq 4.(6)$. The possible minimal configurations for $FFDL(post_p)$ are: R_3X, Z_2Y, YYX, YX_bX, $XXXX$, and thus $w_{alg}(post_p) \geq 3.(6)$, and the statement to prove holds.

Suppose $w_{opt}(pre_p) = 3$. $busy_{alg}(pre_p) \geq 2$, and the statement to prove holds.

Suppose $w_{opt}(pre_p) = \{3.1(6), 3.2(7), 3.(3), 3.5, 3.(5)\}$. $busy_{alg}(pre_p) \geq 2.(3)$. There are no Z_1-tasks in pre_p by Lemma C.7. If there is a Z_2- or greater task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 2.5$. Else there must be a Y-task in $FFDL(pre_p)$, or $pre_p \geq 4$, and $w_{alg}(pre_p) \geq 3$, since 2 X-tasks can not fill a time of $pre_p - 1 = 3$. Then $w_{alg}(pre_p) \geq 2.5$. Concluding $w_{alg}(pre_p) \geq 2.5$, and since when $OPT(pre_p)$ has 2 U_{cp}-tasks in it $w_{opt}(post_p) < 2$, (1) holds.

Suppose $w_{opt}(pre_p) = \{3.(6), 3.(7), 3.9(4), 4, 4.(1), 4.1(6), 4.2(7), 4.(3), 4.(4), 4.5, 4.(5), 4.(6)\}$. The possible minimal configurations of $FFDL(pre_p)$ are: R_4, R_3, Z_2X, YY, YXX, XXX, and $w_{alg}(pre_p) \geq 3$. If $w_{opt}(pre_p) \geq 4$, $w_{opt}(post_p) < 2$, and $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.(3)$. Thus the statement to prove holds when $w_{opt}(pre_p) \leq 4.(3)$. If $w_{opt}(pre_p) \geq 4.5$, the minimal optimal configurations are R_3XX,R_4X, Z_2XX, $U_{cp}XXX$, YX_bXX, $YXXX$, Z_1YX, Z_1XXX, $XX_bX_bX_b$ $w_{opt}(post_p) = 0$, and $w_{alg}(post_p) - w_{opt}(pre_p) \geq 1.(6)$, unless $OPT(pre_p) = YXXX$, in which case $w_{opt}(pre_p) \leq 1$. Then $w_{alg}(post_p) - w_{opt}(pre_p) \geq 1.5$, and (1) holds unless $OPT(pre_p) = YX_bXX$, and $OPT(post_p) = X_a$. Then we have $busy_{alg}(pre_p) = 3.8(3)$. $busy_{alg}(post_p) \geq 1+1+\epsilon \geq 2.(6)$. The minimal configurations of $FFDL(post_p)$ are Z_2, YX, X_bX, and $w_{alg}(post_p) \geq 2.1(6)$, and $w_{alg}(p) \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_p) \in \{4.7(2), 5, 5.0(5), 5.1(6), 5.(3), 5.5, 5.(6)\}$. The optimal configuration may be $U_{cp}U_{cp}X_b$ (and $U_1 > 1.8(3)$),$U_{cp}U_{cp}Y$ (and $U_1 > 1.8(3)$), $U_{cp}U_{cp}U_{cp}$, $XXXXX$, $X_bX_bX_bX$, $YYXX$, X_bXXXX, X_bX_bXXX, YX_bXXX. In all possible cases $busy_{opt}(pre_p) \geq 5$, and thus $busy_{alg}(pre_p) > 4$, and there is no task in $OPT(post_p)$. The minimal configurations of $FFDL(pre_p)$ are R_5, R_4, R_3X, Z_2Y, Z_2X, YYX, YXX, $X_bX_bX_b$, $X_bX_aX_a$ and $p = p^{**}$, $X_aX_aX_aX_a$. Thus, unless $p = p^{**}$, $w_{alg}(pre_p) \geq 3.5$, and else $w_{alg}(pre_p) = 3.1(6)$. Thus the statement to prove holds if $w_{opt}(pre_p) \in \{4.7(2), 5, 5.0(5), 5.1(6)\}$. Suppose $w_{opt}(pre_p) \in \{5.(3), 5.5, 5.(6)\}$. If there are 3 U_{cp}-tasks in $OPT(post_p)$, $busy_{opt}(post_p) > 3 * 1.8(3) = 5.5$, and when the optimal configuration is $YXXXX$ or X_bX_bXXX, and any other configuration with this weight we also have $busy_{opt}(pre_p) \geq 5.5$. Then $busy_{alg}(pre_p) > 4.5$. The minimal configurations of $FFDL(pre_p)$ are R_5, R_4, R_3X, Z_2Y, Z_2XX, YYX, YXX and $p = p^*$, $X_bX_bX_bX$, $XXXX$, and $w_{alg}(pre_p) \geq 4$ unless $p = p^*$, and we have the statement to prove in all cases.

Suppose there are 2 pretimes on p. Assuming without loss of generality that $w_{opt}(pre_{1p}) \geq w_{opt}(pre_{2p})$, we must have $w_{opt}(pre_{1p}) \leq 3.(6)$.

Suppose $w_{opt}(pre_{1p}) = 3.(6)$. Then $w_{alg}(pre_{1p}) \geq 3$, since the same argument from above, when we considered $w_{alg}(pre_p) = 3.(6)$, applies. Also $w_{opt}(pre_{2p}) \leq 2$, and $w_{alg}(pre_{2p}) \geq 1.(6)$. Since $busy_{opt}(pre_{1p}) + pre_{2p} \geq 5$, no task fits in $OPT(post_p)$, and we have $w_{alg}(p) \geq 3 + 1.(6) + 1.(6) = 6.(3) \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) \in \{3.1(6), 3.(3), 3.5\}$. Then $w_{alg}(pre_{1p}) \geq 2.5$, since the same argument from above, when we considered $w_{alg}(pre_p) \in \{3.1(6), 3.(3), 3.5\}$, applies. Also $w_{opt}(pre_{2p}) \leq 2.5$, and $w_{alg}(pre_{2p}) \geq 1.(6)$. Since $busy_{opt}(pre_{1p}) + pre_{2p} \geq 5$, no task fits in $OPT(post_p)$, and we have $w_{alg}(p) \geq 2.5 + 1.(6) + 1.(6) = 5.8(3) \geq w_{opt}(p)$, since, if $w_{opt}(pre_{1p}) \in \{3.5, 3.(3)\}$ $w_{opt}(pre_{2p}) \leq 2.1(6)$.

Suppose $w_{opt}(pre_{1p}) = 3$. $busy_{alg}(pre_p) \geq 2$. $pre_{1p} < 4$, thus there are Y-tasks available to be scheduled in $FFDL(pre_{1p})$. The minimal configurations for $FFDL(pre_{1p})$ are R_3, Z_2, YX, and $w_{alg}(pre_{1p}) \geq 2.5$. $w_{opt}(pre_{2p} \leq 2.(6)$, and we have $w_{alg}(p) \geq 2.5 + 1.(6) + 1.(6) \geq 3 + 2.(6) \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) \in \{2.(6), 2.(7)\}$, and there is a U_{cp}-task in $OPT(pre_{1p})$. Then $busy_{alg}(pre_{1p}) > U_1$, and no single task < 2 can be in $FFDL(pre_{1p})$. We have $w_{alg}(pre_{1p}) \geq 2$, and $w_{alg}(p) \geq 2 + 2w(U_1) \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) = 2.(6)$, and $OPT(pre_p) = YX_b$. $busy_{alg}(pre_{1p}) > 1.8(3)$, and, if $U_1 > 1.8(3)$, $w_{alg}(pre_p) \geq 1.(7)$ and if $w_{opt}(post_p) = 0$ we have $w_{alg}(p) \geq$

$5.(3) \geq w_{opt}(p)$ while if $w_{opt}(post_p) = 1$ we have $w_{alg}(p) \geq 1.(7) + 1.(7) + 2.5 \geq 6.0(5) \geq 5.(6) = w_{opt}(p$. Else (if $U_1 \leq 1.8(3)$), $w_{alg}(pre_p) \geq 2$, and if $w_{opt}(post_p) = 1$ we have $w_{alg}(p) \geq 2 + 1.(6) + 2.5 \geq w_{opt}(p)$, and else there is no task in $OPT(post_p)$ and $w_{alg}(p) \geq 5.(3) \geq w_{opt}(p)$. Recall that $w_{opt}(pre_{2p} \leq w_{opt}(pre_{1p})$.

Suppose $w_{opt}(pre_{1p}) = 2.5$. If $w_{opt}(pre_{2p}) = 2.5$ no task is in $OPT(post_p)$, and (1) holds. If $w_{opt}(pre_{2p}) = 2.25$ there may be an X_a task in $OPT(post_p)$. Then $w_{alg}(post_p) \geq 2.5$, and $w_{opt}(p) \geq 5.8(3) > 5.75 \geq w_{opt}(p)$. If $w_{opt}(pre_{2p}) \leq 2$, we have $w_{opt}(post_p) \leq 1.25$, and $w_{opt}(p) \geq 5.8(3) > 5.75 \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) = 2.25$. $w_{opt}(pre_{2p}) \leq 2.25$. $w_{opt}(post_p) \leq w(U_1)$. Then $w_{alg}(post_p) \geq 2.5$. If there is no task in $OPT(post_p)$, $w_{alg}(p) \geq 5 > w_{opt}(p)$. Else if there is a U_1-task in $OPT(post_p)$, we have $busy_{alg}(post_p) \geq U_1 + 1 + \epsilon > 3$. A Z_2-task can not fill this time by itself. If there is a U-task U_4 in $FFDL(post_p)$, $U_4 \leq U_1$, and at least another U-task or two X-tasks must also be in $FFDL(post_p)$. No two X-tasks can fill a time of $U_1 + 1 + \epsilon$, thus, if there are only X-tasks in $FFDL(post_p)$ we have $w_{alg}(post_p) \geq 3$, which is true in all other cases as well. Thus $w_{alg}(p) \geq 3 + 2w(U_1) \geq 4.(6) + w(U_1) > 4.5 + w(U_1) \geq w_{opt}(p)$. Suppose $w_{opt}(post_p) \in \{1.25, 1.5\}$. Then $busy_{opt}(pre_{1p}) + busy_{opt}(post_p) \geq 4.(6)$, and since $opt < 6$, $w_{opt}(pre_{2p}) \leq 2$. $w_{alg}(p) \geq 2.5 + 1.(6) + 1.(6) = 5.8(3) > 5.75 \geq w_{opt}(p)$. Suppose $w_{opt}(post_p) = 1$. Then $w_{alg}(p) \geq 2.5+1.(6)+1.(6) > 1+2.25+2.25 \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) = 2 \geq w_{opt}(pre_{2p})$. If there is a U_1-task in $OPT(post_p)$, as shown in the previous case we have $w_{alg}(post_p) \geq 3$. Then $w_{alg}(p) \geq 3.(3) + 3 > 4 + w(U_1) \geq w_{opt}(p)$. Else if $w_{opt}(post_p) \in \{1, 1.25, 1.5\}$, $w_{alg}(p) \geq 2.5 + 3.(3) = 5.8(3) \geq 4 + 1.5 \geq w_{opt}(p)$. Else $w_{alg}(p) \geq 5 > w_{opt}(p)$. $\triangle$

Theorem C.11

If $opt \geq 5$, then statement (s3) of Theorem C.9 does not hold.

Proof: Suppose $opt \geq 5$ and statement (s3) of Theorem C.9 holds. Then there is a time slot $ts \geq 5$ in the FFDL-schedule of which there is at least one task $X_1 \in (\frac{4}{3}, 1+\epsilon]$ and one other task $\in [1, 1+\epsilon]$. This implies in that every time slot $ts_1 < 5$, tasks that are longer than 4/3 fill a time that is $> ts_1 - X_1$. Also all time slots in which there is place for a task $\in (4/3, 1+\epsilon]$ but a task $\in [1, 4/3]$ is scheduled are considered after ts and have a minimum weight that is $\geq w_{min}(5)$. We shall refer to the previous statement as (a2).

We shall consider the weights used in the proof of Theorem C.9, with the following exceptions. X_b-tasks are X-tasks the length of which is in the range $(\frac{4}{3}, 1+\epsilon]$, and their weight is 1.25. U_{cp}-tasks are U-tasks that are equal to the U-tasks scheduled by FFDL on the compensating processor, and are weighed at 1.(6), if they are $< 1.(6)$, and at 1.75 otherwise. Z_{12}-tasks are Z-tasks in the range $(2.(3), 2+\epsilon]$, and their weight is 2.25. R_{32}-tasks are R_3-tasks in the range $(3+\epsilon, 4)$, and weigh 3.5.

We also use all other notations used in the proof of Theorem C.9. Also, X-tasks that are not X_b-tasks will be called X_a-tasks. Z_1-tasks that are not Z_{12}-tasks will be called Z_{11}-tasks. U-tasks that are not U_{cp}-tasks will be called U_a-tasks. R_{31}-tasks are R_3-tasks that are $\leq 3+\epsilon$.

Let p^{**} be the processor on which the last X_b-task was scheduled by FFDL and p^* the processor on which the last U-task was scheduled by FFDL. We show that for all $p \in P$, $p \notin \{p^{**}, p^*\}$, we have (1): $w_{opt}(p) \leq w_{alg}(p)$, and that $w_{alg}(p) \geq w_{opt}(p) - 0.25$ if $p \in \{p^*, p^{**}\}$ and $p^* \neq p^{**}$, and $w_{alg}(p) \geq w_{opt}(p) - 0.5$ if $p = p^* = p^{**}$. This is a contradiction, since it implies $\sum_{p\in P} w_{alg}(p) \geq \sum_{p\in P} w_{opt}(p) - 0.5$, when in truth $\sum_{p\in P} w_{alg}(p) \geq \sum_{p\in P} w_{opt}(p) - 1$, since all tasks occur in the optimal schedule and $\overline{X}$ does not occur in the FFDL-schedule.

Let p be an arbitrary processor.

Since the new task types have a weight that is greater than that it had in the proof of Theorem C.9, we have $w_{alg}(p) \geq w_{opt}(p)$ unless there is at least an X_b-task or a U_{cp}-task in $OPT(p)$ or the fact that p fulfills statement (s3) or statement (s2) was used in the proof.

Statement (s3) was used in the proof when we had $w_{opt}(pre_p) = 5$, there were only 3 X-tasks in $FFDL(pre_p)$, and at least one of them was an X_b-task.

Then, if there is only one X_b-task X_2 in $FFDL(pre_p)$, we have $p = p^{**}$, since any other X_b-task would fit in the remaining time $pre_p - X_2 > 5 - (1+\epsilon)$, and we have $w_{opt}(p) - 0.25 \leq w_{alg}(p)$. Else, there are two or more X_b-tasks on p, and we have $w_{alg}(p) \geq 3 + 0.5 + 1.5 \geq 5 = w_{opt}(p)$. We have also shown that

$$w_{min}(5) \geq 3.25.$$

Suppose there are 2 pretimes on p. There can not be a U-or greater task in $OPT(post_p)$, since then $opt > 1 + \epsilon + 4 \geq opt$.

Suppose there is an X_b-task is in $OPT(post_p)$. Then we have $busy_{alg}(post_p) > 2 + \epsilon$, and $post_p \geq 3 + \epsilon$. Suppose $post_p \geq 5$. Then $w_{alg}(post_p) \geq w_{min}(5) \geq 3.25$. Suppose $post_p < 5$. Then, by (s3) there are enough X_b-tasks left to be scheduled on p, and since one X_b-task can fill a time of at most $1+\epsilon$, there is enough space for two on them in $post_p \geq 3+\epsilon$. Then we have $w_{alg}(post_p) \geq 2.5$. The same is true when there is a Z_2-task or a U-task in $FFDL(post_p)$. Since $w_{opt}(pre_{1p}) + w_{opt}(pre_{2p}) \leq 4.25$, as no 3 X_b-tasks and any combination of tasks weighing at least 2 can fit in any optimal schedule. Neither can a combination of tasks weighing at least 4 and a U-task do that (else $busy_{opt}(p) > 4 + \frac{1}{2}opt - 1 > opt$). Then we have $w_{opt}(p) \leq 5.5 \leq 1.(6) + 1.(6) + 2.5 \leq w_{alg}(pre_{1p}) + w_{alg}(pre_{2p}) + w_{alg}(post_p) = w_{alg}(p)$.

Suppose there is an X_a-task in $OPT(post_p)$. The same argument as above can be used to show that $w_{alg}(post_p) \geq 2.5$. $w_{opt}(pre_{1p}) + w_{opt}(pre_{2p}) \leq 4.5$, and

$w_{alg}(p) \geq 5.8(3) > w_{opt}(p)$.

Suppose there is no task in $OPT(post_p)$. If the weights of both pretimes are ≤ 2, we have (1). Let $i, j \in \{1, 2\}$ such that $w_{opt}(pre_{ip}) \geq w_{opt}(pre_{jp})$.

Suppose $w_{opt}(pre_{ip}) \leq 2.25$. Then we have $w_{alg}(p) \geq 3 * 1.(6) = 5 \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{ip}) = 2.5$, and there are 2 X_b-tasks in $OPT(pre_{ip})$. We have $w_{opt}(pre_{jp}) \leq 2.5$, and since $w_{alg}(p) \geq 5$, (1) holds.

Suppose there is a U-task and an X-task in $OPT(pre_{ip})$. If it is an X_a-task we have $w_{opt}(pre_{ip}) = 2.5$, and (1) holds. If it is an X_b-task we have $busy_{opt}(pre_{ip}) \geq 1.(3) + 1.5$. Suppose there is a single task that is < 2 in $FFDL(pre_{ip})$. Then $U_1 > 1.8(3)$ and $opt \geq busy_{opt}(cp) > 5.5$. Then $\epsilon = 1/2opt - 2 \geq 1.75$. Then $busy_{opt}(pre_{ip}) \geq 1.75 + 1.(3) > 3$, and $busy_{alg}(pre_p) > 2$, contradiction. Thus $w_{alg}(pre_{ip}) \geq 2$. Then $w_{alg}(p) \geq 2 + 5/3 + 5/3 = 5 + 1/3 \geq 2.75 + 2.5 \geq w_{opt}(p)$. If there is a U and an X-task in $OPT(pre_{ip})$, then there can not be a U and an X-task or a Z_2-task in $OPT(pre_{jp})$, thus $w_{opt}(pre_{jp}) \leq 2.5$, as the most weight is created when there are 2 X_b-tasks in the optimal schedule of that time slot.

Suppose $w_{opt}(pre_{ip}) = 1.(6) + 1.25 = 2.91(6)$, or $w_{opt}(pre_{ip}) = 1.75 + 1.25 = 3$, and there is a U_{cp}-task and an X_b-task in $OPT(pre_{ip})$. We have $w_{alg}(pre_{ip}) \geq 2$. Also $w_{opt}(pre_{jp}) \leq 2.25$, and $w_{alg}(pre_{jp}) = w(U_1) = w_{alg}(post_p)$. In both cases we have $w_{alg}(p) \geq 2 + 5/3 + 5/3 = 5 + 1/3 > 5.25 \geq w_{opt}(p)$.

Suppose there is only one pretime, pre_p on p. We have shown above that if $w_{opt}(pre_p) \leq 2.5$, we have $w_{alg}(pre_p) \geq 5/3$, since all arguments apply to pre_p as they applied to pre_{ip} in the previous case. We also showed that, if $w_{opt}(pre_p) \in \{2.75, 2.91(6)\}$, $w_{alg}(p) \geq 2$.

If $w_{opt}(pre_p) = 3$, we have again $w_{alg}(pre_p) \geq 2$.

We next consider $post_p$.

Suppose $w_{opt}(post_p) = 0$. Then $w_{alg}(post_p) \geq w(U_1) \geq 1.(6)$.

Suppose $w_{opt}(post_p) \in \{1, 1.25\}$. Then $busy_{alg}(post_p) \geq 2 + \epsilon$. We have sown above that in this case $w_{alg}(post_p) \geq 2.5$.

Suppose $w_{opt}(post_p) = 1.5$. Then we have $busy_{alg}(post_p) > 1 + \epsilon + 1 + \epsilon \geq 3$. No two X-tasks can fill this time. No Z-task by itself can do that either. Suppose there is a U-task U_4 in $FFDL(post_p)$. If there is only a U-task and an X-task in $FFDL(post_p)$, we have $p = p^*$, and $w_{alg}(post_p) \geq 2.75$, since there must be X_b-tasks available if $post_p$ is considered before ts from statement (s3), and else we have $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) \in \{1.(6), 1.75\}$. Then one U_{cp}-task is in $OPT(post_p)$, and $busy_{alg}(post_p) > U_1 + 1 + \epsilon \geq 3$. A Z_2-task can not fill this time alone. Any U-task U_4 scheduled by FFDL in $post_p$ fulfills $U_4 \leq U_1$, so another U-task or tasks of total weight ≥ 2 are needed to fill $busy_{alg}(post_p) - U_4$, since one X-task is not enough. Suppose there are only X-tasks in $FFDL(post_p)$. There must 3 at least 3 of them. We have $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) = 2$. The argument in the proof of Theorem C.9 holds and we have $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) = 2.25$. $busy_{alg}(post_p) > 2.(3) + 1 + \epsilon$. An R_{31}-task can't fill this time, and neither can a Z_1-task and an X_a-task. A Z_2-task can't fill this time alone. Suppose FFDL scheduled a U-task U_4 in $post_p$. $busy_{alg}(post_p) - U_4 > 1 + \epsilon + 0.(3) \geq 1.8(3)$. Suppose there is a U-task $\geq 1.8(3)$ in $post_p$. Then $U_1 \geq 1.8(3)$, $busy_{opt}(cp) > 3U_1 \geq 5.5$, and $\epsilon > 0.75$. Then $busy_{alg}(post_p) - U_4 > 1 + 0.75 + 0.(3) > 2$, and so there can not be only one other U-task besides U_4 in $post_p$, and we have $w_{alg}(post_p) \geq 3.5$ in this case. Suppose there are only X-tasks in $FFDL(post_p)$. Suppose there are only 3 of them. Suppose $post_p \geq 5$. Then at least one X-task is an X_b-task and we have $w_{alg}(post_p) \geq 3.25$ if $p = p^{**}$, and else $w_{alg}(post_p) \geq 3.5\}$. Else there are still X_b-tasks left according to statement (s3), and we have $w_{alg}(p) \geq 3.5$, as at least two of them fit in $post_p$, no matter how big they are. We have in all cases $w_{alg}(post_p) \geq 3.5$ unless $p = p^{**}$, and if $p = p^{**}$, $w_{alg}(p) \geq 3.25$.

Suppose $w_{opt}(post_p) = 2.5$, and there are 2 X_b-tasks in $OPT(post_p)$. We have $busy_{alg}(post_p) > 2+2/3+1+\epsilon > 4$. An R_3-task can not fill this time. Suppose there is a Z_2-task M_2 in $FFDL(post_p)$. Then we have $busy_{alg}(post_p) - M_2 > 2/3 + \epsilon$, and $w_{alg}(post_p) \geq 3.5$. Suppose there is a Z_1-task M_2 in $FFDL(post_p)$. This can not happen by Lemma C.7, since $post_p > 3 > post_{cp}$. Suppose there is a U-task U_4 in $FFDL(post_p)$. $U_4 < 2$, and any task with length < 2 can not fill $busy_{alg}(post_p) - U_4 > 1+2/3+\epsilon$. If there is another U-task in $post_p$, we have $w_{alg}(post_p) \geq 1.5+1.5+1 = 4$, else $p = p^*$, and if there are X_b-tasks at all we have $w_{alg}(post_p) \geq 3.75$, else we have $w_{alg}(post_p) \geq 3.5$. Suppose there are only X-tasks in $FFDL(post_p)$. If there are 4 of them we have $w_{alg}(post_p) \geq 4$. Suppose that there are only 3 X-tasks in $post_p$. 3 X_a-tasks can not fill $busy_{alg}(post_p)$. Suppose we have 1 X_b-task X_2 among them. Then $busy_{alg}(post_p) - X_2 > 2 + 2/3$, and at least one more X_b-task and another X-task are needed to fill this time. Suppose there are two X_b-tasks and an X_a-task in $FFDL(post_p)$. Then we have $p = p^{**}$ and $w_{alg}(post_p) = 3.5$, else we have $w_{alg}(post_p) = 3.75$. In all cases $w_{alg}(post_p) \geq 3.5$.

Suppose $w_{opt}(post_p) \in \{2.5, 2.(6)\}$, and there are a U-task and an X-task in $OPT(post_p)$. $busy_{alg}(post_p) \geq 3 + 2\epsilon \geq 4$. No Z_1-task can be in this time slot by Lemma C.7. If there is a Z_2-task or an R_3- or greater task in $post_p$ we have $w_{alg}(post_p) \geq 3.5$, since a Z_2- or R_3-task can not fill this time by itself. Two U-tasks can not fill a time greater than 4. So if there are at least 2 U-tasks in $post_p$ we have $w_{alg}(post_p) \geq 4$. If there is only one U-task and X-tasks in $FFDL(post_p)$, we must have at least 2 X-tasks in addition to the U-task, and $w_{alg}(post_p) \geq 3.5$. Suppose there are only X-tasks in $post_p$. If there are 4 of them we have $w_{alg}(post_p) \geq 4$. Else the biggest X-task in $post_p$ must be $> 4/3$, thus it must be an X_b-task. Thus, if

$p = p^{**}$ we have $w_{alg}(post_p) \geq 3.25$, and else we have $w_{alg}(p) \geq 3.5$.

Suppose $w_{opt}(post_p) = 2.75$. There are a U-task and an X_b-task in $OPT(post_p)$, or there is a U_{cp}-task greater than 1.(6) and an X_a-task in $OPT(post_p)$. The same argument as above applies.

Suppose $w_{opt}(post_p) \in \{2.91(6), 3\}$, and there are a U_{cp}-task and an X_b-task in $OPT(post_p)$. $busy_{alg}(post_p) > U_1 + 1.(3) + 1 + \epsilon \geq 4.(3)$. If there is an R_4 or an R_3-task in $FFDL(post_p)$, we have $w_{alg}(post_p) \geq 4$. If there is a Z_2-task M_2 in $FFDL(post_p)$ we have $busy_{alg}(post_p) - M_2 > 1.(3)$, at least an X_b-task is needed to fill this time, and $w_{alg}(post_p) \geq 3.8(3)$. Suppose there is a U-task U_4 in $post_p$. $busy_{alg}(post_p) - U_4 > 1.(3) + 1 + \epsilon$, and if there is another U-task in $FFDL(post_p)$ we have $w_{alg}(post_p) \geq 4$. Else at least 2 X_b or 3 X_a-tasks are needed to fill this time, and again $w_{alg}(post_p) \geq 4$. Suppose there are only X-tasks in $FFDL(post_p)$. If there are 4 of them we have $w_{alg}(post_p) \geq 4$. If there are only 3 of them, there must be an X_b-task X_2 among them. $busy_{alg}(post_p) - X_2 > U_1 + 1.(3)$. There must be at least a second X_b-task, else this time can not be filled. Since this task is $< U_1$, the last X-task must also be an X_b-task a,d we have $w_{alg}(post_p) \geq 4$. Thus in all cases $w_{alg}(post_p) \geq 3.8(3)$.

Suppose $w_{opt}(post_p) = 3$, and there is a combination of tasks of length at least 3 in $OPT(post_p)$. We have $busy_{alg}(post_p) \geq 4 + \epsilon$. If there is an R_3 or an R_4-task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 4$. Else if there is a Z_2-task M_2 in $post_p$, $busy_{alg}(post_p) - M_2 > 1 + \epsilon$, and thus $w_{alg}(post_p) \geq 2.5 + 1.5 = 4$. Suppose there are only U- and X-tasks in $FFDL(post_p)$. Suppose there is a U-task U_4 in $FFDL(post_p)$. Then $busy_{alg}(post_p) - U_4 > 2 + \epsilon$. Then, if $p \neq p^*$ we have $w_{alg}(post_p) \geq 4$. Else U_4 is the last scheduled U-task. Suppose ts from statement (s3) fulfills $ts > post_p$, or that ts is considered by FFDL after it considers $post_p$ when FFDL assigns tasks to time slots. Then there are X_b-tasks available for FFDL to schedule in $post_p$, and

$w_{alg}(post_p) \geq 4$. Suppose $ts \leq post_p$, and FFDL considers ts before $post_p$. Then U_4 or other U-tasks would have been scheduled in ts, a contradiction. Recall that the statement affirms that at least 2 X-tasks are scheduled in ts. Suppose there are no U-tasks in $post_p$. If there are at least 4 X-tasks in $post_p$ we have $w_{alg}(p) \geq 4$. Else at least one X-task X_2 must be an X_b-task. Then we have $busy_{alg}(post_p) - X_2 > 3$, and there must be at least a second X_b-task in $post_p$. Then we have,if $p = p^{**}$, $w_{alg}(p) \geq 3.5$, and else $w_{alg}(post_p) \geq 3.75$. We have also shown that $w_{min}(5+\epsilon) \geq 3.5$.

If there is a pretime on p there can not be two U-tasks in $post_p$, since then $opt \geq pre_p + busy_{opt}(post_p) > opt$. Concluding, if $w_{opt}(post_p) \leq 3$, we have $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.75$, if $p \neq p^{**}$, and $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.5$ if $p = p^{**}$.

Suppose $w_{opt}(pre_p) = 2$. Then $w_{alg}(pre_p) \geq 1.(6)$, and in all situations considered above for $post_p$, that is, whenever $w_{opt}(post_p) \leq 3$, (1) holds. We consider $w_{opt}(post_p) > 3$. Suppose $w_{opt}(post_p) = 3.25$. Then we have $busy_{alg}(post_p) > 3.(3) + 1 + \epsilon$, since no two U-tasks can be in $OPT(post_p)$. We have shown in the previous case that a time of $4 + \epsilon$ can only be filled by tasks that weigh together at least 3.75 (in which case $w_{alg} \geq 1.(6) + 3.75 > 5.25 = w_{opt}(p)$), unless $p = p^{**}$, and there are 2 X_b-tasks and one X_a-task on p, in which case $w_{alg}(post_p) = 3.5$. The statement to prove holds.

Suppose $w_{opt}(post_p) = 3.5$. Then there are 2 X_b-tasks in $OPT(post_p)$, and we have $busy_{alg}(post_p) > 4.(6) + \epsilon > 5$. Also $opt > 3.(6) + 2 = 5.(6)$, and $\epsilon > 0.8(3)$. Thus $w_{alg}(pre_p) = w(U_1) = 1.75$, since in this case we must have $U_1 > 1.(6)$. Again, we have $w_{alg}(post_p) \geq 3.75$ (and $w_{alg}(p) \geq 5.5 = w_{opt}(p)$) unless $p = p^{**}$, and $w_{alg}(post_p) \geq 3.5$, and $w_{alg}(p) \geq w_{opt}(p) - 0.25$ otherwise.

There can not be 3 X_b-tasks in $OPT(post_p)$, as then $opt \geq pre_p + 4 \geq 6$, a contradiction.

Suppose $w_{opt}(pre_p) = 2.25$. We again have $w_{alg}(pre_p) \geq 1.(6)$. Whenever $w_{opt}(post_p) \leq 3$, the statement to prove holds, as then $w_{alg}(post_p) \geq w_{opt}(p) + 0.75$ if $p \neq p^{**}$, and $w_{alg}(post_p) \geq w_{opt}(p) + 0.5$ otherwise.

Suppose $w_{alg}(post_p) = 3.25$. $busy_{alg}(post_p) > 4.(3) + \epsilon$, $opt > 3.(6) + 2 = 5.(6)$, and $\epsilon > 0.8(3)$. Then $busy_{alg}(post_p) > 5.1(6)$. If there is an R_3- or greater task in $FFDL(post_p)$, we have $w_{alg}(post_p) \geq 4$, and (1) holds. If there is a Z_2-task M in $FFDL(post_p)$ we have $busy_{alg}(post_p) - M > 2.1(6)$, and we have $w_{alg}(post_p) \geq 4.5$. There can not be a Z_1-task in $FFDL(post_p)$ Lemma C.7. Suppose there is a U-task U_4 in $FFDL(post_p)$. $busy_{alg}(post_p) - U_4 > 3.1(6)$. If there is another U-task on p, we have $w_{alg}(p) \geq 4$. If there is no other U-task on p, and there are only X_a-tasks in addition to U_4 in $FFDL(post_p)$, statement (s3) can not be true, as there is at least X_b-task and another X-task in a time slot ts. If ts is considered by FFDL before $post_p$, then there would be one or more U-tasks instead of the X-tasks in ts, and else, there must be X_b-tasks left after FFDL schedules tasks in $post_p$, and thus there are X_b-tasks available to be scheduled in $post_p$. Then we have $w_{alg}(post_p) \geq 4$. In all cases, $w_{alg}(post_p) \geq 4$, and (1) holds.

If $w_{opt}(pre_p) \geq 2.25$, we can not have $w_{opt}(post_p) \geq 3.5$, and 2 X_b-tasks and an X-task in $OPT(post_p)$ since then $opt \geq 1 + 1.(3) + 1.(3) + 1.(3) + 1 \geq 6$.

Suppose $w_{opt}(pre_p) = 2.5$, and there are 2 X_b-tasks in $OPT(pre_p)$. We have $w_{alg}(pre_p) \geq 1.75$, since, if there is a single task U_4 in $FFDL(pre_p)$ we must have $U_4 = U_1 > 1.(6)$, and thus $w(U_4) = 1.75$. In this case we have $w_{opt}(post_p) \leq 3$, since otherwise there must be a third X_b-task and tasks of length at least 2 in $OPT(post_p)$, and then $opt \geq pre_p + 3.(3) \geq 6$. Thus the statement to prove holds, as then we have $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.75$ if $p \neq p^{**}$, and $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.5$ if $p = p^{**}$.

Suppose $w_{opt}(pre_p) = 2.5$, and there is a U-task and an X-task in $OPT(pre_p)$. We have $w_{alg}(pre_p) \geq 1.(6)$. There can not be a U-task and an X-task in $FFDL(post_p)$, and neither can 3 X-tasks, and $w_{opt}(post_p) \leq 2.5$. We have shown above that in these cases, $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$, and thus (1) holds.

Suppose $w_{opt}(pre_p) \in \{2.(6), 2.75, 2.91(6), 3\}$. In these cases we have $w_{alg}(pre_p) \geq 2 \geq w_{opt}(pre_p) - 1$, as $busy_{alg}(post_p) > U_1$, and no single task weighing less then 2 can fill this time. (1) holds, since in all relevant cases $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$.

Suppose $w_{opt}(pre_p) = 3.25$, and there is an X_b-task in $OPT(pre_p)$. $busy_{alg}(pre_p) \geq 2.(3)$. There are no Z_1-tasks in this time slot by Lemma C.7. If there is a U-task in pre_p we have $w_{alg}(pre_p) \geq 2.5$. Suppose there are only X-tasks in $FFDL(pre_p)$. If ts from statement (s3) was considered by FFDL after pre_p, there is at least one X_b-task in $FFDL(pre_p)$, and $w_{alg}(pre_p) \geq 2.25$. Otherwise $pre_p \geq 5$ and $w_{alg}(pre_p) \geq 3.25$ We have in all cases $w_{alg}(pre_p) \geq 2.25 \geq w_{opt}(pre_p) - 1$, and (1) holds, since in all relevant cases $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$.

Suppose there are 2 U-tasks in $OPT(pre_p)$. Then we have $w_{opt}(post_p) < 2$, and $w_{alg}(post_p) \geq w_{opt}(post_p) + 1.25$. Suppose $OPT(pre_p)$ is made of 2 U_a-tasks or a U_{cp}-task and a U_a-task, and $w_{opt}(pre_p) \leq 3.25$ Then $busy_{alg}(pre_p) \geq 2$, $w_{alg}(pre_p) \geq 2$, and (1) holds. Suppose there are 2 U_{cp} tasks in $OPT(pre_p)$. Any two X_b-tasks would fit in pre_p, as they would be less than the U_{cp}-tasks. Suppose $pre_p < ts$. Then there are X_b-tasks to schedule when FFDL reaches pre_p, and $w_{alg}(pre_p) \geq 2.5$ if there are 2 X-tasks in pre_p. If there is at least a U-task in pre_p we have $w_{alg}(pre_p) \geq 2.5$, since at least another task fits in pre_p besides the U-task. Note that $pre_p > 3$, thus there is no Z_1-task in pre_p. In all cases $w_{alg}(pre_p) \geq 2.5$, and we have (1), if pre_p is considered before ts by FFDL. Suppose pre_p is considered after ts by FFDL. Then we have $pre_p \geq 5$, and $w_{alg}(pre_p) \geq 3.25$, and (1) holds.

Suppose $w_{opt}(pre_p) = 3.5$, and $OPT(p)$ is not made of 2 U-tasks. Either there is a U-task and 2 X-tasks or there are 2 X_b-tasks and an X_a-task in $OPT(pre_p)$. In the first case we have $busy_{alg}(pre_p) \geq 2 + \epsilon$, and any two X_b-tasks fit in $pre_p \geq 2 + 1 + \epsilon$. If $pre_p < 5 \leq ts$, the relevant minimal configurations are Z_2, UX, X_bX_b, and $w_{alg}(pre_p) \geq 2.5$, and else $w_{alg}(pre_p) \geq 3.25$. (1) holds. In the second case, $OPT(pre_p) = X_bX_b$, we have $busy_{alg}(pre_p) \geq 2.(6)$, and if $FFDL(pre_p)$ is made of 2 X-tasks, at least one of them must be a X_b-task. If there is a U-task or a Z_2- or greater task, or there are 3 X-tasks in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 2.5$, and (1) holds. If there is one X_b-task X_2 in $FFDL(pre_p)$ we have $busy_{alg}(pre_p) - X_2 > X_3$, where X_3 is any of the two X_b-tasks scheduled in $OPT(pre_p)$. Thus if X_2 was not the last X_b-task to be scheduled by FFDL, we have $w_{alg}(pre_p) \geq 2.5 \geq w_{opt}(pre_p) - 1$, and else we must have $pre_p \geq ts \geq 5$, and $w_{alg}(pre_p) \geq 3.25$. In all cases we have (1).

Suppose $w_{opt}(pre_p) \in \{3.(6), 3.75\}$, and there is a U_{cp}-task and tasks the total length of which is at least 2 in $OPT(pre_p)$. We have $w_{opt}(post_p) \leq 1.25$, and $w_{alg}(post_p) \geq w_{opt}(post_p) + 1.25$. $busy_{alg}(pre_p) \geq U_1 + 1$. If there are tasks of type Z_2 or greater in $FFDL(pre_p)$ we have $w_{alg}(pre_p) \geq 2.5$. No task of type Z_1 can be in pre_p. If there is a U-task U_4 in pre_p, $U_4 \leq U_1$, and $w_{alg}(pre_p) \geq 2.5$. Suppose there are only X-tasks in pre_p. If FFDL considers pre_p before ts, there are 2 X_b-tasks in $FFDL(pre_p)$, and $w_{alg}(pre_p) \geq 2.5$. Else $pre_p \geq 5$, and $w_{alg}(pre_p) \geq 3.25$. We have $w_{alg}(pre_p) \geq 2.5$, and the statement to prove holds.

Suppose $w_{opt}(pre_p) \in \{3.75, 4\}$, and there are either 3 X_b-tasks or another combination of tasks of length at least 4 in $OPT(pre_p)$. We have $busy_{alg}(pre_p) \geq 3$. A Z_2-task is not enough to fill this time. If there is a U-task U_4 in $FFDL(pre_p)$, FFDL must have considered pre_p before ts, else U_4 would be scheduled in ts. If there is a second U-task, we have $w_{alg}(pre_p) \geq 3$. If there is no second U-task we have $p = p^*$, since a second U-task would fit in pre_p. Also since any X_b-task would fit in $post_p - U_4$, and FFDL considered pre_p before ts, there must also be an X_b-task in pre_p, and $w_{alg}(pre_p) \geq 2.75$. Suppose there are only X-tasks in pre_p. If there are 3 or more of them we have $w_{alg}(pre_p) \geq 3$, and (1) holds for this and all the previous cases, since $w_{alg}(pre_p) \geq 2.75$, $w_{opt}(post_p) \leq 1.25$, and in then $w_{opt}(post_p) \leq w_{alg}(post_p) - 1.25$. Suppose there are only 2 X-tasks $X_1 \geq X_2$ in $FFDL(pre_p)$. If pre_p was considered by FFDL before ts $w_{alg}(pre_p) = 2.5$ (and else $w_{alg}(pre_p) \geq 3.25$ and there can not be only 2 X-tasks in $FFDL(pre_p)$). We have $pre_p \geq 4$, and thus $X_1 > (pre_p - 1)/2 = 1.5$. Thus, by Theorem C.10 we have $opt < 5.(3)$. Then $busy_{opt}(post_p) < 1.(3)$, thus $w_{opt}(post_p) \leq 1$. Then $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.5$, and (1) holds.

Suppose $w_{opt}(pre_p) = 4.25$. Unless there are only 2 X-tasks in $FFDL(pre_p)$ the same argument as in the previous case holds, and $w_{alg}(pre_p) \geq 3$ unless $p = p^*$, in which case $w_{alg}(pre_p) \geq 2.75$. There can be no U-task in $OPT(post_p)$, thus $w_{opt}(post_p) \leq 1.25$, and we have: if $p = p^*$, $w_{alg}(p) \geq w_{opt}(p) - 0.25$, and else $w_{alg}(p) \geq w_{opt}(p)$. Suppose there are only 2 X-tasks in $FFFL(pre_p)$. Since $busy_{opt}(pre_p) \geq 4.(3)$, there can be no task in $OPT(post_p)$ by Theorem C.10. The X-tasks scheduled in $FFDL(post_p)$ must average $\geq 3.(3)/2 = 1.(6)$, thus $w(U_1) = 1.75$. Then $w_{alg}(p) \geq 4.25 = w_{opt}(p)$.

Suppose $w_{opt}(pre_p) = 4.5$, and there are 2 X_b-tasks and 2 X-tasks in $OPT(pre_p)$. Then $w_{opt}(post_p) \leq 1$, and either $p = p^{**}$, and $w_{alg}(p) \geq w_{opt}(p) + 1.25$, or $w_{alg}(p) \geq w_{opt}(p) + 1.5$. The same argument as in the previous case holds, and we have: if $p = p^* \neq p^{**}$ or if $p = p^{**} \neq p^*$ $w_{alg}(p) \geq w_{opt}(p) - 0.25$, if $p = p^{**} = p^*$ and else $w_{alg}(p) \geq w_{opt}(p) - 0.5$, and else $w_{alg}(p) \geq w_{opt}(p)$.

Suppose $w_{alg}(pre_p) \in \{4.5, 4.(6), 4.75\}$, and there are a U-task and tasks the total length of which is ≥ 3 in in $OPT(pre_p)$. No task can be in $OPT(post_p)$, and $w_{alg}(post_p) \geq w(U_1)$. $busy_{alg}(pre_p) \geq 3 + \epsilon$. If there is a Z_2- or greater task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 3$. If there is a U-task U_4 in $FFDL(pre_p)$ we have

$busy_{alg}(pre_p) - U_4 \geq 1 + \epsilon$, so $w_{alg}(pre_p) \geq 3$ again. Suppose there are only X-tasks in $FFDL(pre_p)$. At least 3 of them must be there to fill a time of $2+1+\epsilon$. We have $w_{alg}(post_p) \geq 3$, and $w_{alg}(p) \geq w(U_1) + 3 \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_p) \in \{4.75, 4.8(3), 5\}$ there is a U-task U_0, an X_b-task, and 2 X_a-tasks or a Z_{11}-task in $OPT(pre_p)$. $busy_{alg}(p) \geq 1+\epsilon+2.(3) \geq 1.5+2.(3) \geq 3.8(3)$. A R_{31}-task or a Z_2-task can not fill this time. If a task of any of those types or a task that is longer than that is in pre_p we have $w_{alg}(pre_p) \geq 3.5$. No Z_1-task can be in $FFDL(pre_p)$. If there is a U-task U_4 in $FFDL(pre_p)$ we have $busy_{alg}(pre_p) - U_4 > 1.(3) + \epsilon \geq 1.8(3)$. If there is a U-task $> 1.8(3)$ in pre_p, then $U_1 > 1.8(3)$, and $opt > 3U_1 > 5.5$, $\epsilon > 0.75$, and $busy_{alg}(pre_p) - U_4 > 1.(3)+\epsilon > 2$, and $w_{alg}(pre_p) \geq 3.5$ if there is a second U-task in pre_p. Else, if there is no second U-task in pre_p, again $w_{alg}(pre_p) \geq 3.5$. Suppose there are only X-tasks in $FFDL(pre_p)$. Then, if $pre_p < 5$, there must be at least 2 X_b-tasks and another X-task in $FFDL(pre_p)$. Then we have $w_{alg}(p) \geq 3.5 + w(U_1) > w_{opt}(p)$ Else if $pre_p \geq 5$, we have from $w_{min}(5) = 3.25$, that $w_{alg}(p) \geq 3.25 + w(U_1) \geq 3.25 + w(U_0) = w_{opt}(p)$.

Suppose $w_{opt}(pre_p) \in \{5, 5.1(6), 5.25\}$, and there are 2 X_b-tasks, a X_a-task and a U-task, or 5 X_a-tasks in $OPT(pre_p)$. We have $pre_p \geq 5$, and $w_{alg}(pre_p) \geq 3.25$ if $p = p^*$, and $w_{alg}(pre_p) \geq 3.5$ otherwise, as shown when we considered $w_{opt}(pre_p) = 5$ at the beginning of this proof. We have $w_{alg}(p) \geq 3.25 + w(U_1) \geq w_{opt}(p) - 0.25$ if $p = p^{**}$, and $w_{alg}(p) \geq w_{opt}(p)$ if $p \neq p^{**}$.

Suppose there are 1 X_b-task and tasks of length at least 4 in $OPT(pre_p)$, and $w_{opt}(pre_p) = 5.25$. $busy_{alg}(pre_p) > 4.(3)$. If there is an R_3- or greater task in $FFDL(pre_p)$, we have $w_{alg}(pre_p) \geq 4$. If there is a Z_2-task M in $FFDL(pre_p)$, $busy_{alg}(pre_p) - M \geq 1.(3)$, at least a X_b-task is required to fill this time, and $w_{alg}(pre_p) \geq 3.75$. If there is a U-task U_4 in $FFDL(pre_p)$ we have $busy_{alg}(pre_p) - U_4 > 2.(3)$. If there is a second U-task in $FDDL(pre_p)$, $w_{alg}(pre_p) \geq 4$. Since there is a U-task in pre_p, FFDL considered pre_p before ts, and there are X_b-tasks left for FFDL to schedule in pre_p: $w_{alg}(pre_p) \geq 3.75$. Suppose there are only X-tasks in pre_p. If there are at least 4 of them we have $w_{alg}(pre_p) \geq 4$, and else at least one of them, X_2, is $> 1.(3)$. Suppose $\epsilon \leq 0.(6)$. Then $busy_{alg}(pre_p) - X_2 > 4.(3) - 1.(6) = 2.(6)$, and so at least one second task in $FFDL(pre_p)$ is an X_b-task. We have, if $p = p^{**}$, $w_{alg}(pre_p) \geq 3.5$, and $w_{alg}(p) > w_{opt}(p) - 0.25$, and else $w_{alg}(pre_p) \geq 3.75$, and $w_{alg}(p) \geq w_{opt}(p)$. If $\epsilon > 0.(6)$, we have $w_{alg}(p) \geq 3.25 + 1.75 \geq w_{opt}(p) - 0.25$ if $p = p^{**}$, and $w_{alg}(p) \geq 3.5 + 1.75 \geq w_{opt}(p)$ if $p \neq p^{**}$.

Suppose $w_{opt}(pre_p) \geq 5.5$, and there are 2 X_b-tasks and tasks of length at least 3 in $OPT(pre_p)$. $busy_{alg}(pre_p) \geq 4.(6)$, $\epsilon > 0.(6)$, and $w(U_1) = 1.75$. If this time slot is considered after ts, there are only X-tasks and possibly R_5-tasks left to schedule for FFDL in pre_p. Suppose there are only X-tasks in pre_p. If there are 4 or more X-tasks in $FFDL(pre_p)$ we have $w_{alg}(pre_p) \geq 4$. If there are only 3 X-tasks in $FFDL(pre_p)$ at least 2 of them must be X_b-tasks, since $busy_{alg}(pre_p) - 2 > 2.(6)$, and any first X_b-task is < 2. We have $w_{alg}(p) \geq 3.5 + 1.75 \geq w_{opt}(p) - 0.25$ if $P = p^{**}$, and $w_{alg}(p) \geq w_{opt}(p)$ if $p \neq p^{**}$. Note that after scheduling 2 X_b-tasks in pre_p, the remaining space is $> 1.(6)$, and that there are X_b-tasks $< 1.(6)$ that can be scheduled in pre_p (if $p \neq p^{**}$), because the X_b-tasks in the optimal schedule of this processor must fulfill this condition, else $busy_{opt}(p) \geq 6$. If pre_p is considered before ts we have, as in the previous case, $w_{alg}(pre_p) \geq 3.75$, and (1).

Suppose there is no pretime on p. We have already considered the cases when $w_{opt}(post_p) \leq 3.5$, and the task configuration in $post_p$ is compatible with adding a time of length 2 without exceeding the length of the optimal schedule.

Next, we consider the case when there are a U-task and two X-tasks, 3 X_b-tasks, or task combinations with the same weight which have not been considered before in $OPT(post_p)$. We have already considered the case when there are 2 X_b-tasks and one X_a-task in $post_p$.

Suppose $w_{opt}(post_p) \in \{3.5, 3.(6), 3.91(6), 3.75, 4\}$ $busy_{alg}(post_p) > 4 + 2\epsilon \geq 5$. If there is an R_4- or an R_3-task in $FFDL(post_p)$, (1) holds. If $post_p$ is considered after ts, there are no Z or U-tasks in $post_p$. Then, at least 4 X-tasks are needed to fill a time $> 4 + 2\epsilon = 2 + (1 + \epsilon) + (1 + \epsilon)$. Suppose $post_p$ is considered by FFDL before ts. If there is a Z_2-task M in $FFDL(post_p)$, $busy_{alg}(post_p) - M > 1 + 2\epsilon \geq 2$, and we have $w_{alg}(post_p) \geq 4.5$. Suppose there is a U-task U_4 in $FFDL(post_p)$. $busy_{alg}(post_p) - U_4 > 2 + 2\epsilon$, and there are also either another U-task and an X_b-task, or 3 X-tasks in $FFDL(post_p)$, and $w_{alg}(post_p) \geq 4.25$. Suppose there are only X-tasks in $FFDL(post_p)$. They are all X_b-tasks, and there are at least 4 of them. We have (1).

Suppose $w_{opt}(post_p) \in \{4.1(6), 4.25, 4.(6), 4.75, 4.8(3), 4.91(6), 5\}$, and there are 2 X_b-tasks and a U_{cp}-task, or a U_{cp}-task and tasks of a total weight of 3 or 3.25 or 3.1(6) in $OPT(post_p)$. The following argument holds for any optimal schedule configuration of $post_p$ containing a U_{cp}-task and any configuration of tasks with length > 2.6 and a total weight that is ≤ 3.25.

$busy_{alg}(post_p) > U_1 + 2.(6) + 1 + \epsilon = U_1 + 3.(6) + \epsilon \geq 5.(6)$. If there is an R_5-task or an R_4-task in $FFDL(post_p)$ we have (1). If there is a R_3-task M_3 in $FFDL(post_p)$ we have $busy_{alg}(post_p) - M_3 > U_1$, and thus $w_{alg}(post_p) \geq 5$, since no single task < 2 is $> U_1$. Else, if $post_p$ is considered after ts we have only X-tasks in $post_p$. If there are at least 5 of them we have $w_{alg}(post_p) \geq 5$. Else, if there are only 4 of them at least 3 must be X_b-tasks, since all X-tasks are $< U_1$ and $< 1 + \epsilon$, and any two X-tasks that add up to more than 2.(6) must have a X_b-task among them. We have again $w_{alg}(p) \geq 5$.

Suppose $post_p$ is considered before ts. Suppose there is a Z_2-task M_2 in $post_p$. $busy_{alg}(post_p) - M_2 > U_1 + 0.(6) + \epsilon$. If there is another Z_2-task there we have

$w_{alg}(post_p) \geq 5$ Else, if there also is U-task in $post_p$, there must be at least another task there, and $w_{alg}(post_p) \geq 2.5 + 1.5 + 1 \geq 5$.

If the maximum task in $post_p$ is a U-task U_4, and there is a second U-task U_5 in $post_p$ we have $busy_{alg}(post_p) - U_4 - U_5 > 1.(6) + \epsilon > 2$, and $w_{alg}(post_p) \geq 5$. If there are only X-tasks in $post_p$, we have, by the same argument as above, $w_{alg}(post_p) \geq 5$. In all cases $w_{alg}(p) \geq 5$, and (1) holds.

We have also shown that $w_{min}(U_1 + 2.(6) + 1 + \epsilon) \geq 5$. If in $OPT(post_p)$ there are two X_b-tasks and tasks with a total weight of 2, or one X_b-task and one U_a-task and tasks with a total weight of 2, or one U_a-task, two X_b-tasks and one X_a-task, we have $post_p > 2 + 2.(6) + 1 + \epsilon > U_1 + 2.6 + \epsilon$, and $w_{alg}(post_p) \geq 5 \geq w_{opt}(post_p)$.

Suppose there are 4 X-tasks such that there is at least one X_b-task among them, or one X_b-task and tasks of weight 3 in $OPT(post_p)$. Then $w_{opt}(post_p) = 4.25$. $busy_{alg}(post_p) > 4.(3) + 1 + \epsilon$. If there is an R_5-task or an R_4-task in $FFDL(post_p)$ we have $w_{alg}(post_p) \geq 5$. If there is a R_3-task M_3 in $FFDL(post_p)$ we have $busy_{alg}(post_p) - M_3 > 1.(3) + \epsilon$. At least a U-task is needed to fill this time and $w_{alg}(post_p) \geq 4.5$. If $post_p$ is considered before ts, and if there is a Z_2-task M in $FFDL(post_p)$, $busy_{alg}(post_p) - M > 2.(3) + \epsilon$. If there is another Z_2-task $w_{alg}(post_p) \geq 5$. If there is a U-task in addition to M, $w_{alg}(post_p) \geq 2.5 + 1.5 + 1 = 5$, If there are only two X-tasks, they must be X_b-tasks, and $w_{alg}(post_p) \geq 2.5 + 1.25 + 1.25 \geq 5$. If the maximum task in $FFDL(post_p)$ is a U-task U_4, $busy_{alg}(post_p) - U_4 > 2.(3) + 1 + \epsilon$, and to fill this time a weight of at least 3 is needed, thus $w_{alg}(post_p) \geq 4.5$. If there are only X-tasks in $FFDL(post_p)$, and there are no more than 4 of them, the average length of an X-task is $> 1.458(3)$. Thus at least one of them is an X_b-task, and $w_{alg}(p) \geq 4.25$.

Suppose there are 3 U_{cp}-tasks in $OPT(post_p)$, and $U_1 \leq 1.(6)$. $w_{opt}(post_p) = 5$, and since two U_{cp}-tasks have a total length $> 2.(6)$, and $w_{min}(U_1 + 2.(6) + 1 + \epsilon) \geq 5$, we have $w_{alg}(post_p) \geq 5 \geq w_{opt}(p)$.

If $U_1 > 1.(6)$, and there are 3 U_{cp}-tasks in $OPT(post_p)$, $busy_{opt}(post_p) \geq 5$. $busy_{alg}(post_p) \geq 5+1+\epsilon \geq 6.(6)$. The same is true when there is a U_{cp}-task, two X_b-tasks, and one X_a-task, or a U_a-task, two X_b-tasks and one X_a-task in $OPT(post_p)$. Then $busy_{alg}(post_p) \geq 1.5+3.(6)+1+\epsilon \geq 6.(6)$. This also holds when there are tasks of a total weight of 3 and 2 X_b-tasks in $OPT(post_p)$, and $busy_{alg}(post_p) > 6.5+\epsilon \geq 7$, and $w_{opt}(post_p) = 5.5$, and when there are tasks of total weight 4 and one X_b-task in $OPT(post_p)$, and $busy_{alg}(post_p) \geq 5.(3) + 1 + \epsilon \geq 6.8(3)$.

Note that no U-task and 4 X-tasks can be scheduled in $OPT(post_p)$, thus we can not have $w_{opt}(post_p) = 5.1(6)$.

In the above described cases $w_{opt}(post_p) \in \{5, 5.25, 5.5\}$. We have $busy_{alg}(post_p) > 5 + 1 + \epsilon$. If there is an R_5-task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 6$. If there is an R_4-task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 5.5$. If there is an R_{32}-task M_3, $busy_{alg}(post_p) - M_3 > 2 + \epsilon$, and $w_{alg}(post_p) \geq 5.5$. If there is a R_{31}-task M_3, $busy_{alg}(post_p) - M_3 > 3$, and at least one X_b-task and one other X-task, or a U-task and one other task, or a Z-task and one other task, or another R_3-task are needed to fill this time. Then $w_{alg}(post_p) \geq 5.25$ if $p = p^{**}$, and $w_{alg}(post_p) \geq 5.5$ otherwise.

If there is a Z_2-task M_2 in $FFDL(post_p)$, $busy_{alg}(post_p) - M_2 > 3 + \epsilon$, and $w_{alg}(post_p) \geq 5.5$.

If the maximum task in $FFDL(post_p)$ is a U-task U_4, $busy_{alg}(post_p) - U_4 > 4+\epsilon$. If there is a second U-task U_5 in $FFDL(post_p)$, $busy_{alg}(post_p) - U_4 - U_5 > 2+\epsilon$, and if there is a third U-task, $w_{alg}(post_p) \geq 5.5$. If there is no third U-task, but there are U-tasks in $FFDL(post_p)$, $post_p$ must have been considered by FFDL before ts, and there are X_b-tasks available for FFDL to schedule in $post_p$. Then $w_{alg}(post_p) \geq 5.5$. Suppose there is no second U-task. Then there must be at least 3 X_b-tasks in $FFDL(post_p)$ in addition to U_4, and $w_{alg}(post_p) \geq 5.5$. Suppose there are only X-

tasks in $FFDL(post_p)$. If there are 6 or more of them, (1) holds. If there are 5 of them, at least one is $> (6+\epsilon)/5 > 6.(6)/5 = 4/3$. Then $w_{alg}(p) \geq 5.25$ if $p = p^{**}$, and $w_{alg}(p) \geq 5.5$ otherwise.

If there are only 4 X-tasks in $FFDL(post_p)$, their average is $> (busy_{opt}(p) + 1 + \epsilon)/4$. Suppose $opt < 5.(3)$. Then $w_{opt}(post_p) = 5$, and since at least one task in $FFDL(post_p)$ is a X_b-task, and the other ones average at a time length $> 5/3 = busy_{alg}(post_p) - 1 - \epsilon$, there are at least 2 X_b-tasks in $FFDL(post_p)$. So, either $p = p^{**}$, and $w_{alg}(post_p) \geq 4.5 \geq w_{opt}(p) - 0.5$, or $w_{alg}(post_p) \geq 5 \geq w_{opt}(post_p)$.

Suppose $w_{opt}(post_p) \in \{5.25, 5.5\}$. No U-task and 4 other tasks can be in $OPT(post_p)$. We have $busy_{opt}(post_p) \geq 5.(3)$, and by Theorem C.10 there can not be any X-tasks greater than 1.5 in $FFDL(post_p)$. $busy_{alg}(post_p) \geq 5.(3)+1+0.(6) = 7$. If there were only 4 or less X-tasks in $FFDL(post_p)$, their average would be > 1.75, contradicting Theorem C.10, thus there can not be only 4 X-tasks in $FFDL(post_p)$. We have shown above that in all other cases the statement to prove holds. $\triangle$

Theorem C.12 ((s2) does not hold)

Statement (s2) of Theorem C.9 does not hold if $opt \geq 5$.

Proof: We use the weights and notations from Theorem C.9 with the following exceptions: there are X_b-tasks in the range $(1+\epsilon/2, 1+\epsilon]$ weighed at 1.25 if they are < 1.75, with the subcategory X_c-tasks in the range $[1.75, 1+\epsilon]$ weighed at 1.(3), U-tasks $\in (1+\epsilon, 2)$ weighed at 1.5 unless they are equal to U_1, with the subcategory U_{cp}-tasks of size U_1 which are weighed at 1.(6) if they are $\leq 1+\epsilon+\epsilon/2$, and at 1.75 otherwise. Z_2-tasks$\in (2+\epsilon, 3]$ are weighed at 2.5, R_{32}-tasks$\in (3+\epsilon, 4]$ weighed at 3.5, and R_{42}-tasks $\in [4+\epsilon, 5)$, weighed at 4.5. We shall call X_a-tasks X-tasks that are not X_b-tasks, Z_1-tasks Z-tasks that are not Z_2-tasks, and R_{31}-tasks R_3-tasks that are not R_{32}-tasks.

Let p^{**} be the processor on which the last X_b-task is scheduled by FFDL. Let p^* be the processor on which the last X_c-task is scheduled by FFDL. We show that, unless $p = p^{**}$ or $p = p^*$, $w_{alg}(p) \geq w_{opt}(p)$, and otherwise $w_{alg}(p) \geq w_{opt}(p) - 0.5$ if $p = p^{**} \neq p^*$, and $w_{alg}(p) \geq w_{opt}(p) - 0.25$ if $p = p^* \neq p^{**}$, and $w_{alg}(p) \geq w_{opt}(p) - 0.75$ if $p = p^* = p^{**}$.

Statement (s2) implies that:

(a3) there are no tasks in the range $(1+\epsilon, 3+\epsilon]$ in the FFDL-schedule of time slots which are greater than $3+2\epsilon$, which follows from the scheduling policy.

Suppose there is no pretime on p.

Suppose $w_{opt}(post_p) = 0$. Then $w_{alg}(post_p) \geq 1.5$.

Suppose $w_{opt}(post_p) \in \{1, 1.25, 1.(3)\}$, and there is a X-task in $post_p$. Then $busy_{alg}(post_p) \geq 2+\epsilon$. If there are only 2 X-tasks in this time slot then at least one of them is a X_b-task, and either $p = p^{**}$ and $w_{alg}(post_p) \geq 2.25$ or $w_{alg}(post_p) \geq 2.5$. If there is a U- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 2.5$.

Suppose $w_{opt}(post_p) \in \{1.5, 1.(6), 1.75, 2\}$. $busy_{alg}(post_p) > 2+2\epsilon$, and $post_p > 3+2\epsilon$. Two X-tasks can not fill this time. U-task or a Z-task can not be in $FFDL(post_p)$ by statement (a3). Thus $w_{alg}(post_p) \geq 3$.

Suppose $w_{opt}(post_p) \in \{2.25, 2.(3), 2.5, 2.(6), 2.8(3), 3, 3.08(3), 3.25, 3.5\}$ Here, we assume the weight of 3.25 is obtained from a U_{cp}-task and a U_a-task, and the weight of 3.5 is obtained from 2 U_{cp}-tasks. In all possible cases $busy_{opt}(post_p) \geq 2+\frac{1}{2}\epsilon$. $busy_{alg}(post_p) > 3(1+\frac{1}{2}\epsilon)$. Thus, if there are only 3 X-tasks in $FFDL(post_p)$, at least one of them is a X_b-task, and we have: if $p = p^{**}$ $w_{alg}(post_p) \geq 3.25$, and else $w_{alg}(post_p) \geq 3.5$.

In case $w_{opt}(post_p) \geq 2.5$, we have $busy_{opt}(post_p) \geq 2+\epsilon$, $busy_{alg}(post_p) \geq 3+2\epsilon$, and $post_p \geq 4+2\epsilon = 2+(1+\epsilon)+(1+\epsilon)$. Then any three X_b-tasks fit in $post_p$, and we have either $w_{alg}(post_p) \geq 3.75$, or $p = p^{**}$, and $w_{alg}(post_p) \geq 3.25$ in these cases.

Suppose $w_{opt}(post_p) \in \{3.25, 3.5, 3.(3)\}$. $busy_{alg}(post_p) > 3+\epsilon/2+1+\epsilon$. Suppose 3 X-tasks can fill this time. Then one of them has to be $> 4/3 + \epsilon/2$, and so it is an X_b-task. The other two tasks need to add up to $3 + \epsilon/2$, so at least a second one must be an X_b-task, which actually it is an X_c-task, since it is $> 1.5 + \epsilon/4 \geq 1.75$. The X_c-task is $< 1 + \epsilon$, and so the other two tasks need to add up to $> 3 + \epsilon/2$, implying that there is a second X_c-task. Then, if $p = p^{**}$, $w_{alg}(post_p) \geq 3.(6)$, and else $w_{alg}(post_p) \geq 3.8(3)$.

If p is not the processor with the last X_c-task on it we have $w_{alg}(post_p) \geq 4$. Suppose $OPT(post_p)$ has the configuration $X_cX_aX_a$, and there is a pretime on p. Then $opt > 5.(6)$, $\epsilon > 0.8(3)$, $post_p > 3.(6) + 1 + 0.8(3) > 4.(6) + \epsilon$. $post_p - X_c = 2 + X_{c1}$, where X_{c1} is the X_c-task the optimal schedule has in $post_p$. Either a task $\leq X_{c1}$ was already scheduled on $FFDL(p)$ and any other two X_c-tasks fit, or X_{c1} and any other task fits in the remaining time. Therefore, if $w_{opt}(post_p) = 3.25$, either $p = p^*$, the last processor with a X_c-task scheduled on it, and $w_{alg}(post_p) \geq 3.(6)$ (2 X_c-tasks and a X_a-task), if p is also p^{**}, or $w_{alg}(post_p) \geq 3.91(6)$ if p is not also p^{**}, and else $w_{alg}(post_p) \geq 4$.

Suppose there is a U_{cp}-task, a U-task and an X-task in $OPT(post_p)$, and that $w_{opt}(post_p) \in \{4, 4.25, 4.(3), 4.5, 4.58(3), 4.(6), 4.75\}$ Note that $OPT(post_p)$ can not be made of one X_c-task and two U_{cp}-tasks if $U_1 > 1 + \epsilon + \epsilon/2$, since then $busy_{opt}(post_p) \geq 2U_1+1.75 > 3.75+2\epsilon+2(\epsilon/2) > 4+\epsilon = opt$. We have $busy_{alg}(post_p) > U_1 + 1 + \epsilon + 1 + 1 + \epsilon \geq 4 + 3\epsilon \geq 5 + 1/2\epsilon$. If there is a R_{32}- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 4.75$. If there are only X-tasks in $FFDL(post_p)$ there must be at least 4 of them, since 3 of them can not add up to $U_1 + 2 + (1 + \epsilon) + \epsilon$. If there are more than 4 tasks $w_{alg}(post_p) \geq 5$. Since $busy_{alg}(post_p) > 4 + 2\epsilon$, at least one of the four tasks is a X_b-task. Thus, either $p = p^{**}$, and $w_{alg}(post_p) \geq 4.25$, or $p \neq p^{**}$, and $w_{alg}(post_p) \geq 4.75$, as any three X_b-tasks fit in a time of length

$(1+\epsilon)+(1+\epsilon)+2$.

Recall that there can not be a U-task and any combination of tasks of length at least 4 in $OPT(post_p)$, else $busy_{opt}(post_p) > opt$.

Suppose $w_{opt}(post_p) \in \{4, 4.1(6), 4.25, 4.(3), 4.41(6), 4.5, 4.(6), 4.75\}$, and there is no U_{cp}-task in $OPT(post_p)$ except when the weight is 4.1(6) and 4.41(6). We have $busy_{alg}(post_p) > 4+1+\epsilon$. If there is a R_{32}- or greater task in $FFDL(post_p)$ we have $w_{alg}(post_p) \geq 5$. Else we have only X-tasks. 3 or less of them add up to $< 2+2+1+\epsilon$. If there are 4 of them, one of them has a length $> 1+0.25+\epsilon/4$. We know that $\epsilon < 1$, and thus $0.25 > eps/4$. Then at least one of the tasks is a X_b-task. Thus, if $p = p^{**}$, $w_{alg}(post_p) \geq 4.25$, and else $w_{alg}(post_p) \geq 4.75$.

Suppose there are 3 U_{cp}-tasks in $OPT(post_p)$. Then $busy_{alg}(post_p) > 4(1+\epsilon) \geq 6$. If there is a R_{32}- or greater task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 5.5$. If there are only X-tasks there must be at least 5 of them, since all X-tasks are $< 1+\epsilon$, and $w_{alg}(post_p) \geq 5$. Thus if $U_{cp} < 1+\epsilon+\frac{eps}{2}$ the statement to prove holds. Suppose $U_1 > 1+\epsilon+\epsilon/2$. Then $busy_{alg}(post_p) \geq 3+3\epsilon+\frac{3}{2}\epsilon \underset{\epsilon>0.5}{\geq} 4+\epsilon+\frac{3}{2}\epsilon > opt$, a contradiction.

We can not have the optimal configuration $XX_cX_cX_c$, since the busy time would add up to 6. Neither is $X_bX_bX_bX_b > 4+2\epsilon = opt$ possible. We treated the cases when the weight adds up to 4.75 and 4.(6) (configurations $X_aX_bX_bX_b$, and $X_aX_aX_cX_c$ in $post_p$) previously.

Suppose the configuration of $OPT(post_p)$ is $X_aX_bX_cX_c$ or $X_aX_bX_bX_c$. In this case $\epsilon > 0.75$, and $w_{opt}(post_p) = \{4.91(6), 4.8(3)\}$. $busy_{alg}(post_p) \geq 4+\frac{3}{2}\epsilon+1+\epsilon \geq 5+\frac{5}{2}\epsilon$. If there is a R_{32}, a R_4 or R_5-task in $post_p$, we have $w_{alg}(post_p) \geq 5$. Else there are only X-tasks in $FFDL(post_p)$. Suppose there are only 4 of them. Then at least one of them, X_1, is a X_b-task since $busy_{alg}(post_p) > (1+\epsilon)+(1+\epsilon)+2+(1+\epsilon/2)$. We have $X_1 < 1+\epsilon$, and at least a second one X_2, is a X_b-task. $X_1+X_2 < 2+2\epsilon$, and at least a third one, X_3, must be a X_b-task. $X_3 < 2$, and the last task must be a X_b-task as well. $w_{alg}(post_p) \geq 5$.

Suppose $w_{opt}(post_p) = 5$. $busy_{alg}(post_p) > 6 + \epsilon$. If there is a R_{32}-task or a R_4-task in $FFDL(post_p)$, $w_{alg}(post_p) \geq 5.5$. Else there are only X-tasks in $FFDL(post_p)$. Three of them can not add up to more than 6. If there are only 4 of them, at least one of them must be a X_b-task, since their average is $> 1.62 > 1 + \frac{1}{2}\epsilon$. The first X_b-task is $< 1 + \epsilon$, so the other 3 add up to more than 5. Then at least one of them must be greater than 5/3, and so it is a X_b-task. The last two tasks add up to > 3. Thus at least one of them must be > 1.5, and thus be a X_b-task. Thus, unless $p = p^{**}$, we have $w_{alg}(post_p) = 5$, and otherwise we have $w_{alg}(post_p) \geq 4.75$.

Suppose $w_{opt}(post_p) \in \{5.25, 5.(3)\}$. Then $\epsilon > (1 + \frac{1}{2}\epsilon)/2$, and $\epsilon > 2/3$. Also, $busy_{alg}(post_p) \geq 5 + \frac{1}{2}\epsilon + 1 + \epsilon = 6 + \frac{3}{2}\epsilon$. If there is a R_{32}, R_4 or R_5-task in $post_p$, we have $w_{alg}(post_p) \geq 5.5$, since $busy_{alg}(post_p) - 5 \geq 1 + 3/2\epsilon)$. If there are only 4 X-tasks in $FFDL(post_p)$ their average is $> 1.5 + \frac{3}{8}\epsilon \geq 1.75$. Thus at least one of the tasks X_{c1} is a X_c-task. If $p \neq p^*$, $w_{alg}(post_p) \geq 5.(3) \geq w_{opt}(post_p)$, since $post_p \geq 8$ and any 4 X-tasks fit in this time slot. We have $busy_{alg}(post_p) - X_{c1} \underset{\epsilon > 0.75}{>} 7.125 - 2 = 5.125$. Then the average of the remaining tasks is > 1.70, and there must be at least another X_b-task X_2, in $FFDL(post_p)$. We then have $busy_{alg}(post_p) - X_{c1} - X_2 > 5.125 - 1.75 = 3.375$, and thus there must also be a third X_b-task in $FFLD(post_p)$. Subtracting 1.75 from 3.375, we conclude that the fourth task is also a X_b-task, and $w_{alg}(post_p) \geq 5.08(3)$. Suppose there are 5 X-tasks in $FFDL(post_p)$. Then their average is > 1.425. Suppose $\frac{busy_{alg}(post_p)}{5} < 1 + \epsilon/2$. Then $\frac{6 + \frac{3}{2}\epsilon}{5} < 1 + \epsilon/2$, and $6 + \frac{3}{2}\epsilon < 5 + 5\epsilon/2$. Then $1 < \epsilon$, contradiction. Thus there is at least one X_b-task in $FFDL(post_p)$. Then, if $p = p^{**}$, $w_{alg}(post_p) \geq 5.25$, and else $w_{alg}(post_p) \geq 6.25$. Concluding, if $p = p^*$, we have $w_{alg}(p) \geq 5.08(3) \geq w_{opt}(p) - 0.25$, if $p = p^{**}$, we have $w_{alg}(p) \geq 5.25$, and else $w_{alg}(post_p) \geq 5.(3) \geq w_{opt}(post_p)$.

Suppose there is a pretime on p. Then there can not be 2 U-tasks in $OPT(post_p)$, else $pre_p + busy_{opt}(post_p) > opt$.

Suppose $w_{opt}(pre_p) \in \{2, 2.25\}$. Then $w_{alg}(pre_p) \geq 1.(6)$. $w_{opt}(post_p) \leq 3.(3)$, and there are no 2 U-tasks in $FFDL(post_p)$ we have $w_{alg}(post_p) \geq w_{opt}(pre_p) + 0.5$, unless $p = p^{**}$ or $p = p^{*}$, and else $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.25$). Then $w_{opt}(p) \leq w_{alg}(p)$ if $p \neq p^{**}$, and $w_{opt}(p) - 0.25 \leq w_{alg}(p)$ otherwise.

Suppose $w_{opt}(pre_p) \in \{2.25, 2.(3)\}$. Then $w_{opt}(post_p) \leq 3.08(3)$, since there can not be 2 X_b-tasks and 3 other tasks in $OPT(p)$ (else $busy_{opt}(p) \geq 5 + \epsilon > opt$). $w_{alg}(pre_p) \geq 1.(6)$, and the statement to prove holds, since in these cases $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.(6)$, or $p = p^{**}$ and $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.1(6)$.

Suppose $w_{opt}(pre_p) = \{2.5, 2.(6)\}$. Then $busy_{opt}(pre_p) \geq 2 + \epsilon$. Then there can not be a U-task and an X-task or two X_b-tasks in $OPT(post_p)$. We have $w_{alg}(pre_p) \geq 1.(6)$. Whenever $w_{opt}(post_p) \leq 2.(3)$, we have $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$.

Suppose $w_{opt}(pre_p) = 2.75$, and there is a U-task and an X_b-task in $OPT(pre_p)$. Then, if there is a single task $U_4 < 2$ in $FFDL(pre_p)$, we must have $U_4 = U_1 \geq 1 + \epsilon + \frac{1}{2}\epsilon$, and thus $w(U_4) = 1.75$, and $w_{alg}(p) \geq w_{opt}(p)$.

Suppose there is a U_{cp}-task and an X-task in $OPT(pre_p)$, and $w_{opt}(pre_p) \in \{2.75, 2.91(6), 3, 3.08(3)\}$. There are no Z_1-tasks in the FFDL-schedule of time slots $> 2 + \epsilon$. If there is a U-task in $FFDL(pre_p)$, then there is at least another task in that time slot. If there are only 2 X-tasks in $FFDL(pre_p)$, either $pre_p < 3 + \epsilon$, and both X-tasks are X_b-tasks, or $pre_p \geq 3 + \epsilon$, and at least one X-task is an X_b-task, and then we have $w_{alg}(pre_p) \geq 2.25$. The statement to prove holds, since in all cases $w_{alg}(post_p) \geq w_{opt}(post_p) + 1$.

Suppose there are only 2 U-tasks in $OPT(pre_p)$. $w_{opt}(pre_p) \in \{3, 3.1(6), 3.(3), 3.5\}$. There can be either no task or one X-task in $OPT(post_p)$, and thus $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.91(6)$ if the last X_b-task is scheduled in $FFDL(post_p)$, and $w_{alg}(post_p) \geq$

$w_{opt}(post_p) + 1.1(6)$ otherwise. Also $w_{alg}(pre_p) \geq 2.25$ if $p = p^{**}$, and $w_{alg}(pre_p) \geq 2.5$ otherwise. The statement to prove holds,

Suppose there are one X-task and two X_a-tasks or a Z_1-task, or a U_a-task and 2 X_a-tasks or a Z_1-task, or two X_b-tasks which are not X_c-tasks and one X_a-task in $OPT(pre_p)$. $w_{opt}(pre_p) \in \{3, 3.25, 3.(3), 3.5\}$. The same argument as in the previous case holds in this case as well.

Suppose there are one X_b-task, one X_c-task and another X-task in $OPT(pre_p)$. Then $w_{opt}(pre_p) \in \{3.58(3), 3.(6), 3.91(6), 4\}$. $busy_{opt}(pre_p) > 1.75 + 1.375 + 1 = 4.125$. $busy_{alg}(pre_p) > 3.125$, and $busy_{alg}(pre_p) > 2 + \epsilon$. Since there is at most one X-task in $FFDL(post_p)$, $w_{alg}(post_p) \geq w_{opt}(post_p) + 0.91(6)$ if the last X_b-task is scheduled in $FFDL(post_p)$, and $w_{alg}(post_p) \geq w_{opt}(post_p) + 1.1(6)$ otherwise.

If there is a U- or greater task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 2.5$. Otherwise, either there are 3 or more X-tasks in $FFDL(pre_p)$, and $w_{alg}(pre_p) \geq 3$, or there must be at least one X_b-task in that time slot. If it is not an X_c-task there must be at least a second X_b-task in $FFDL(pre_p)$, and $w_{alg}(pre_p) \geq 2.5$. Else, either $p = p^* = p^{**}$, and the last X_b-task coincides in this case with the last X_c-task and is scheduled by FFDL in pre_p, and $w_{alg}(pre_p) \geq 2.(3)$, or $w_{alg}(post_p) \geq 2.58(3)$. The statement to prove holds for the cases $w_{opt}(pre_p) \in \{3.58(3), 3.(6)\}$.

Suppose we have $w_{opt}(pre_p) \in \{3.91(6)\}$, and there are 2 X_c-tasks in $OPT(pre_p)$. $busy_{alg}(pre_p) \geq 3.5$. Also, $busy_{opt}(post_p) < 1.5$, and so there is not enough place for an X_c-task in $OPT(post_p)$. then $w_{alg}(post_p) - w_{opt}(post_p) \geq 1$ if the last X_b-task is scheduled by FFDL in $post_p$, and $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.25$ otherwise. If there is a Z_2- or greater task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 3$. If there is a U-task U_4 in $FFDL(pre_p)$, $busy_{alg}(pre_p) - U_4 > 1.5$, and $w_{alg}(pre_p) \geq 2.75$. Suppose there are only X-tasks in $FFDL(pre_p)$. If there are 3 of them the statement to prove holds. Suppose there are only 2 of them. then at least one of them is an X_c-task, since both of them add up to a time that is ≥ 3.5. If $p \neq p^*$, $w_{alg}(pre_p) \geq 2.(6)$, and (1) holds. If $p = p^*$, the second task in $FFDL(pre_p)$ must be an X_b-task, and $w_{alg}(pre_p) \geq 2.58(3)$, and the statement to prove holds again.

Note that 3 X_c-tasks have a time length that is ≥ 5.25, and a weight of 4. When we shall handle the case of $w_{opt}(pre_p) = 5$, we shall also include this case.

Next we handle the cases when there is a U_{cp}-task and 2 X-tasks or a Z-task, or 2 X_b-tasks and one X-task, or 3 X_b-tasks in $OPT(pre_p)$.

Suppose $w_{opt}(pre_p) \in \{3.(6), 3.75, 3.91(6), 4\}$. $busy_{alg}(pre_p) > U_1 + 1$. $pre_p > 3+\epsilon$. If there is a U-task U_4 in $FFDL(pre_p)$, there is also an X-task in $FFDL(pre_p)$. If there is a R_3-task, we have $w_{alg}(pre_p) \geq 3$. Else there must be either three or more X-tasks in pre_p, or at least one of the two tasks is a X_b-task. Then, if $p = p^{**}$, $w_{alg}(pre_p) \geq 2.25$, and else $w_{alg}(pre_p) \geq 2.5$. In either case, the statement to prove holds, as, if the last X_b-task is not scheduled by FFDL in $post_p$ and $w_{opt}(post_p) > 3.75$, $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.5$, as no X_b-task can be in the optimal schedule of $post_p$ if $w_{opt}(pre_p) \geq 3.75$, and if $w_{opt}(pre_p) = 3.(6)$, $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.1(6)$. If the last X_b-task is scheduled by FFDL in $post_p$ we have $w_{alg}(p) \geq w_{opt}(p) - 0.25$.

Suppose there is a U-task and 2 X_b-tasks in $OPT(pre_p)$. $w_{opt}(pre_p) \in \{4.1(6), 4.25, 4.(3), 4.41(6)\}$. No X_b-task fits in $OPT(post_p)$, thus $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.5$, unless the last X_b-task was scheduled by FFDL in $post_p$, in which case $w_{alg}(post_p) - w_{opt}(post_p) \geq 1.25$. Also, $busy_{alg}(pre_p) > 1 + \epsilon + 1 + \frac{\epsilon}{2} + \frac{\epsilon}{2} = 2 + 2\epsilon \geq 3$. No two X-tasks or Z_2-task can fill this time. Suppose there is a U-task U_4 in $FFDL(pre_p)$. If there is a U_{cp}-task in $OPT(pre_p)$, $busy_{alg}(pre_p) - U_4 > 1 + \epsilon$, and $w_{alg}(pre_p) \geq 3$ in all cases.

Suppose there is a U_a-task and 2 X_b-tasks in $OPT(pre_p)$ $w_{opt}(pre_p) \in \{4, 4.08(3), 4.1(6)\}$. No two X-tasks or Z_2-task can fill this time. Suppose there is a U-task U_4

in $FFDL(pre_p)$. If there is a second U-task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 3$. Else there must be the biggest X_b-task in $FFDL(pre_p)$. If there are no X_c-tasks at all, $w_{opt}(pre_p) = 4$, and $w_{alg}(pre_p) = 2.5$, and the statement to prove holds. Else $w_{alg}(pre_p) \geq 3.8(3)$, and the statement to prove holds again.

Suppose there are 2 U_{cp}-tasks in pre_p. $w_{opt}(pre_p) \in \{3.(3), 3.5\}$. Then we have $w_{alg}(pre_p) \geq 2.5$ if $p \neq p^{**}$, and else $w_{alg}(pre_p) \geq 2.25$. The statement to prove holds, as $w_{opt}(post_p) < 2$.

Suppose the optimal schedule of pre_p is contains 2 U-tasks or tasks of length at least 3 with weight 3 and one X-task. Then we may have $w_{opt}(pre_p) \in \{4, 4.25, 4.(3), 4.1(6), 4.41(6), 4.5\}$. $busy_{alg}(pre_p) > 2 + 2\epsilon$, and at least 3 X-tasks are needed to fill this time. There can not be any U-tasks in $FFDL(pre_p)$ by theorem C.10. $w_{alg}(pre_p) \geq 3 = w_{opt}(pre_p) - 1.5$, and (1) holds.

Suppose there are 2 U-tasks and one X_c-task in $OPT(pre_p)$, and $w_{opt}(pre_p) > 4.5$. $w_{opt}(pre_p) \in \{4.58(3), 4.(6), 4.75, 4.8(3)\}$, and in both cases there are 2 U_{cp}-tasks and one X_c-task in $OPT(pre_p)$. No task can be in $OPT(post_p)$, thus $w_{alg}(post_p) - w_{opt}(post_p) \geq w(U_1)$. $busy_{alg}(pre_p) > 2U_1$. We have, as in the previous case, $w_{alg}(pre_p) \geq 3$, and (1) holds if $w(U_1) = 1.(6)$.

Else, if there is a X_c-task in $OPT(pre_p)$ we have $busy_{alg}(pre_p) > 2 + 3\epsilon + 0.75 > 3.25 + 2\epsilon$. If there is an R_{32}- or greater task in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 4$. Else, if there is a U-task U_4 in $FFDL(pre_p)$, $busy_{alg}(pre_p) - U_4 > 1 + 3/2\epsilon + 0.75 \geq 1.25 + 2\epsilon \underset{\epsilon > 0.75}{>} 2.75$, and $w_{alg}(post_p) \geq 3.5$. Suppose there are only X-tasks in $FFDL(post_p)$. Two X-tasks can't fill a time > 4. If there are only 3 of them, since $busy_{alg}(pre_p) > (1+\epsilon) + (1+\epsilon) + \epsilon + 0.75$, we conclude that the smallest among them must be $> 0.75 + \epsilon \geq 1.5 > 1 + \epsilon/2$. Thus all 3 X-tasks are X_b-tasks, and $w_{alg}(post_p) \geq 3.75$. If there is no X_c-task in $OPT(pre_p)$, $w_{opt}(pre_p) = 4.75$. Also $w_{alg}(post_p) \geq 1.75$, and $w_{opt}(post_p) = 0$. Then, since $w_{alg}(pre_p) \geq 3$, $w_{alg}(p) \geq$

$w_{opt}(p)$.

Suppose there are 2 X_c-tasks and another X-task in $FFDL(pre_p)$. Then we have $busy_{opt}(pre_p) \geq 4.5$. $w_{opt}(pre_p) \in \{3.(6), 3.91(6), 4\}$ $w_{opt}(post_p) < 1.(3)$ since no U-task or X_c-task fits in $OPT(post_p)$. $busy_{alg}(pre_p) > 3.5$. If there is a U-task U_4 in $FFDL(pre_p)$, $busy_{alg}(pre_p) - U_4 > 1.5$, and either another U-task or the biggest X-task, in our case an X_c-task is scheduled in $FFDL(pre_p)$ in addition to U_4, and $w_{alg}(pre_p) \geq 2.8(3)$, and the statement to prove holds. If there are only X-tasks in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 1.(3) + 1.25 = 2.58(3) \geq w_{opt}(pre_p) - 1.08(3)$, in case $w_{opt}(pre_p) = 3.(6)$. If $w_{opt}(pre_p) \in \{3.91(6), 4\}$, there also is an X_b-task in addition to the two X_c-tasks in $OPT(pre_p)$, and $busy_{alg}(pre_p) \geq 3.5 + \epsilon/2 \geq 3.875$. If there are only 2 X-tasks in $FFDL(pre_p)$, $1 + \epsilon > (3.5 + \epsilon/2)/2$. Then $3\epsilon/4 > 1.75 - 1 = 0.75$, and $\epsilon > 1$, contradiction. thus there can not be only 2 X-tasks in $FFDL(pre_p)$, and $w_{alg}(pre_p) \geq 3$.

Suppose there are 4 X-tasks in $OPT(pre_p)$, and there is at most one X_b-task among these. We have $w_{alg}(pre_p) \geq 2.5$, and the statement to prove holds in the case $w_{opt}(pre_p) = 4$. Suppose $w_{opt}(pre_p) = 4.24$. Then $busy_{alg}(pre_p) = 3 + \frac{1}{2}\epsilon$. This time can only be filled by 2 X_b-tasks, the average of which is > 1.5, contradiction Theorem C.10. If there are more than 2 X-tasks or 2 tasks, or a U-task and an X_b-task or a Z_2- or greater task in $FFDL(pre_p)$, we have $w_{alg}(pre_p) \geq 3$, and the statement to prove holds.

Suppose there is a U-task and tasks of lengths between 3 and 4, with weights 3, 3.25, 3.(3), or an equivalent configuration of tasks (such as a Z_2-task or two X_b-tasks and tasks of a total length between 2 and 3) in $OPT(pre_p)$.
$w_{opt}(pre_p) \in \{4.5, 4.75, 4.8(3), 4.(6), 4.91(6), 5, 5.08(3)\}$. Note that a U_{cp}-task weighed at 1.75 is not possible if there are X_c-tasks, else $U_1 > 1 + 0.75 + 0.375 > 2$. $busy_{alg}(post_p) > 1 + \epsilon + 2$. 2 X-tasks can't fill this time, so $w_{alg}(post_p) \geq 3$.

If there is no X_b-task in $OPT(pre_p)$,$w_{opt}(pre_p) \in \{4.5, 4.(6), 4.75\}$, and the statement to prove holds, since $w_{alg}(post_p) - w_{opt}(post_p) \geq w(U_1)$.
If there is an X_b-task in $OPT(pre_p)$, $busy_{alg}(pre_p) > 1 + \epsilon + 2 + \frac{1}{2}\epsilon$, and there must be at least one X_b-task among the 3 X-tasks in $FFDL(post_p)$. If there is no X_c-task in $OPT(post_p)$, or the U-task is not a U_{cp}-task, the statement to prove holds again. If the last X_b-task is scheduled in $FFDL(pre_p)$, $w_{alg}(pre_p) \geq 3.25$, and else we have $w_{alg}(pre_p) \geq 3.75$. Also, $w_{alg}(post_p) \geq w(U_1)$, while $w_{opt}(pre_p) \leq w(U_1) + 3.25$.

Suppose $w_{opt}(pre_p) \in \{5, 5.25, 5.(3)\}$, and there are no 3 U-tasks in $OPT(pre_p)$. We can not have $w_{alg}(pre_p) \geq 5.5$, since then $busy_{alg}(pre_p) \geq 5+\epsilon \geq 3+\frac{1}{2}opt > opt$. $pre_p > 5$. $busy_{alg}(pre_p) > 4$. Suppose there are 3 X-tasks in $FFDL(pre_p)$. This can not happen by Theorem C.11. Else the statement to prove holds, as the relevant minimal configurations for $FFDL(pre_p)$ are R_5, R_4, $R_{32}X$, $XXXX$.

Suppose there are 2 pretimes on p. We assume $w_{opt}(pre_{1p}) \geq w_{opt}(pre_{2p})$ for convenience. There is no loss of generality since the pretimes are interchangeable. Recall that there can not be two U-tasks and two X-tasks in any optimal schedule of one processor.

If $w_{opt}(pre_{1p}) > 3.(3)$, we must have $busy_{opt}(pre_{1p}) + pre_{2p} \geq 3+\epsilon+2 \geq (2+\epsilon)+3 > opt$, a contradiction. Suppose $w_{opt}(pre_{1p}) = 3.(3)$. If $w_{opt}(pre_{2p}) \geq 2.25$, we have $busy_{opt}(p) \geq 3.75+2+\epsilon/2 \geq 6 > opt$, a contradiction. Then we have $w_{opt}(p) = 5.(3)$. $busy_{alg}(pre_{1p}) \geq 2.75 > 2+\epsilon/2$. $pre_{1p} < 3+\epsilon$, else $opt \geq pre_{1p} + pre_{2p} > 5+\epsilon > opt$, and thus there are X_b-tasks available for FFDL to schedule from statement (s2). If there is a U-task or a Z_2- or greater task in $FFDL(post_p)$, we have $w_{alg}(pre_{1p}) \geq 2.5$. If there are only X-tasks in $FFDL(pre_{1p})$ we again have $w_{alg}(pre_{1p}) \geq 2.5$, since there are two X_b-tasks that fit in this time slot (for example the X_b-task scheduled in the optimal schedule and another X_b-task), and FFDL schedules the bigger tasks first. $w_{alg}(p) \geq 2.5 + 1.(6) + 1.(6) = 5.8(3) > w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) = 3.25$. $w_{opt}(pre_{2p}) < 2.25$, since otherwise $busy_{opt}(p) > 3 + \epsilon/2 + 2 + \epsilon/2 = 5 + \epsilon > opt$. The same argument as in the previous case holds.

If $w_{opt}(pre_{1p}) = 3.08(3)$, there is a U_{cp}-task and an X_c-task in $OPT(pre_{1p})$. Then $w_{alg}(pre_{1p}) \geq 2$, $w_{alg}(p) \geq 2 + 1.75 + 1.75 = 5.5 \geq w_{opt}(p)$, since $w_{opt}(p) < 2.(3)$.

Suppose $w_{opt}(pre_{1p}) = 3$. $w_{opt}(pre_{2p}) \leq 2.(3)$, else $opt \geq 5+\epsilon > opt$. $w_{alg}(pre_{1p}) \geq 2$, since $busy_{alg}(pre_{1p}) > pre_{1p} - 1 \geq 2$, and $w_{alg}(p) \geq 2 + 1.(6) + 1.(6) = 5.(3) \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) \in \{2.58(3), 2.(6), 2.75\}$. Note that if there are 2 X_c-tasks or an X_b-task and an X_c-task in $OPT(pre_{1p})$ we have $pre_{1p} \geq 1.75 + 1.(3) > 3$. Since U-task greater than U_1 can be in $FFDL(pre_{1p})$ alone if the configuration of $OPT(pre_{1p})$ is $U_{cp}X$, $w_{alg}(pre_{1p}) \geq 2$. Also, in all cases $busy_{opt}(pre_{1p}) > 2 + \epsilon$, and thus $w_{opt}(pre_{2p}) \leq 2.(3)$. $w_{alg}(p) \geq 2 + 1.(6) + 1.(6) = 5.(3) \geq w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) = 2.5$. $busy_{opt}(pre_{1p}) \geq 2 + \epsilon$, thus $w_{opt}(pre_{2p}) \leq 2.(3)$, and there can be no task in $OPT(post_p)$. $w_{alg}(p) \geq 5 > w_{opt}(p)$.

Suppose $w_{opt}(pre_{1p}) \in \{2.(3), 2.25, 2\}$, or $w_{opt}(pre_{1p}) < 2$. If there is no task in $OPT(post_p)$, $w_{alg}(p) \geq 5 > w_{opt}(p)$, since we assumed $w_{opt}(pre_{1p}) \geq w_{opt}(pre_{2p})$. Else we must have $w_{alg}(post_p) \geq 2.25$ since $busy_{alg}(post_p) > 2 + \epsilon$. Then $w_{alg}(p) \geq 2.5 + 1.(6) + 1.(6) \geq 5.58(3) \geq w_{opt}(p)$. We have $w_{opt}(p) \leq 1.25 + 1.(3) + 3 = 5.58(3)$ since no three X_b-tasks or two X_c-tasks can be in an optimal schedule with 5 tasks. Also note that there can not be a U-task or more than one task on $OPT(post_p)$, since then $opt \geq pre_{1p} + pre_{2p} + 1 + \epsilon > 5 + \epsilon > opt$. △

From Theorems C.5, C.8, C.9, C.11, and C.12, it results that there is no minimal counterexample with at most two downtimes on each processor and in this case we have $C_{max}(Multifit) \leq \frac{3}{2}opt$ if the parameter ϵ of FFDL Multifit is chosen to be less then half the time unit used in the specification of the problem instance. With this, the proof of Theorem A.2 is completed.

CHAPTER V

CONCLUSION

A. Summary

In this thesis we have presented algorithms for scheduling on multiple processors in the presence of machine shutdowns. We considered the case when there is only one shutdown on each machine and presented an LPT-based algorithm the schedule of which ends within 3/2 of the time needed by the optimal schedule or of the end of the last downtime. This bound is tight in the class of polynomial algorithms assuming that $P \neq NP$. LPT and its derivate algorithms give a better balance of tasks on processors than FFD and Multifit-based algorithms, which first fill one processing space before moving on to the next one, but the worst-case bound of Multifit-based algorithms is better in many situations.

The second result presented in this work concerns scheduling on uniform processors with at most one shutdown time on each machine and the aim of minimizing the maximum completion time. LPT has been shown to have a tight bound higher than $\frac{3}{2}$ even for scheduling problems without availability constraints, and thus was not a likely candidate for an optimal algorithm for the class of polynomial algorithms for this case. We considered a variant of Multifit, which first orders the time slots created by the assigning of the deadline (times the processor's speed ratio) in increasing order, and only then uses FFD to assign tasks to time slots. We showed this algorithm finishes within $\frac{3}{2} + \epsilon$ (where ϵ can be chosen by the user) the end of the optimal schedule or the end of the last downtime, which is optimal in the class of polynomial algorithms for same-speed processors, and thus is optimal for the more general case of uniform processors as well.

Last we considered scheduling on same-speed processors with possibly multiple downtimes on each machine. We used the same algorithm as in the previous case (except that there is no need to compensate for variable speed ratios when ordering the time slots), and showed that its schedule finishes within $\frac{3}{2}$ the optimal schedule's end or the end of the last downtime when there are at most 2 downtimes on each machine, and within $(\frac{3}{2} + \frac{1}{2k})$ of the same in case there are more than 2 downtimes on at least one machine.

B. Methods of argument

To prove the upper bound results several methods were used. A classic method is to define and prove the existence of a *minimal counterexample* assuming that the upper bound is broken, and then show that it can not exist.

In the cases we considered, a minimal counterexample had multiple properties common to all three situations. First, the task that the algorithm scheduled last in the case of LPTX, and which FFDL Multifit first was unable to schedule when a deadline at or above the bound was assigned, was the smallest task in the task set of a minimal counterexample. In the case of LPTX, this task is the only one LPTX can not schedule within 3/2 the optimal schedule length or the end of the last downtime. This allowed for normalization of all time lengths to the length of this task.

Second all idle times were smaller than this last task.

In Chapters III and IV we showed that in small pretimes with only one task in the optimal schedule and in the FFDL-schedule, the task in the FFDL-schedule is greater than or equal to that in the optimal schedule. Also, in Chapter IV we showed that all tasks less than 2 scheduled alone in a time slot of length between 2 and 3 by the FFDL- algorithm are equal in a minimal counterexample.

The concept of *task density* allowed us to make arguments about the weight in time slots of any given length. The concept of *minimal configuration* allowed for collapsing arguments which would have taken a paragraph without it, into a row.

Most of each of the proofs is based on the existence of a *compensating processor*, a processor that has more processing time in the optimal schedule than in the schedule of our algorithm without the last task. Proving that such a compensating processor can not exist led in each case to the completion of the proofs.

REFERENCES

[1] Soo Y. Chang and Hark-Chin Hwang, "The worst-case analysis of the multifit algorithm for scheduling nonsimultaneous parallel machines," *Discrete Applied Mathematics*, vol. 92, pp. 135 – 147, June 1999.

[2] D.K. Friesen and M.A. Langston, "Bounds for multifit scheduling on uniform processors," *SIAM Journal on Computing*, vol. 12, pp. 60 – 69, February 1983.

[3] Michael R. Garey and David S. Johnson, "'Strong' NP-Completeness results: Motivation, examples, and implications," *Journal of the ACM*, vol. 25, pp. 499 – 508, July 1978.

[4] Michael R. Garey and David S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*, Bell Laboratories, Murray Hill, New Jersey, 1979.

[5] R.L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. of Applied Mathematics*, vol. 17, pp. 416 – 429, March 1969.

[6] Hark-Chin Hwang and Soo Y. Chang, "Parallel machines scheduling with machine shutdowns," *Computers and Mathematics with Applications*, vol. 36, pp. 21 – 31, June 1998.

[7] Hark-Chin Hwang, Kangbook Lee, and Soo Y. Chang, "The effect of machine availability on the worst-case performance of LPT," *Discrete Applied Mathematics*, vol. 148, pp. 49 – 61, April 2005.

[8] E. G. Coffman Jr., M. R. Garey, and D.S.Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM J. on Computing*, vol. 7, pp. 1–17, February 1978.

[9] C. Y. Lee, L. Lei, and M. Pinedo, "Current trends in deterministic scheduling," *Annals of Operations Research*, vol. 70, pp. 1 – 41, April 1997.

[10] C.Y. Lee, "Parallel machine scheduling with nonsimultaneous machine available time," *Discrete Applied Mathematics*, vol. 30, pp. 53 – 61, January 1991.

[11] C.Y. Lee, "Machine scheduling with an availability constraint," *Journal of Global Optimization*, vol. 9, pp. 395 –416, December 1996.

[12] C. Sadfi, B. Penz, C. Rapine, J. Blazewicz, and P. Formanovicz, "An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints," *European Journal of Operations Research Operations Research*, vol. 161, pp. 3 – 10, February 2005.

[13] E. Sanlaville and G. Schmidt, "Machine scheduling with availability constraints," *Acta Informatica*, vol. 35, pp. 795 – 811, September 1998.

[14] M. Scharbrodt, A. Steger, and H. Weisser, "Approximability of scheduling with fixed jobs," *Journal of Scheduling*, vol. 2, no. 6, pp. 267 – 284, November 1999.

[15] G. Schmidt, "Scheduling with limited machine availability," *European Journal of Operational Research*, vol. 121, pp. 1 – 15, February 2000.

[16] Z. Tan and Y. He, "Optimal online algorithm for scheduling on two identical machines with machine availability constraints," *Information Processing Letters*, vol. 83, pp. 323 – 329, September 2002.

[17] H. Yong, "Uniform machine scheduling with machine available constraints," *Acta Mathematicae Applicatae Sinica (English Series)*, vol. 16, pp. 122 – 129, April 2000.

[18] Minyi Yue, "On the exact upper bound of the multifit processor scheduling algorithm," *Annals of Operations Research*, vol. 24, pp. 233 – 259, December 1990.

VITA

Liliana Gentiana Alex Grigoriu graduated from the Theoretical High-School Hermann Oberth in Bucharest, and received her Diplom in Computer Science from the Technical University of Berlin in December 2000. There she was a member of the Theoretical Computer Science – Formal Specification group. She had a stipend from the Friedrich-Naumann Foundation and was part of the the Daimler-Chrysler fellowship program Research and Technology. She received a Ph.D. in Computer Science from Texas A&M University in May 2010.

Ms. Grigoriu may be reached at Department of Computer Science and Engineering, Texas A&M University, TAMU 3112, College Station, TX 77843-3112, c/o Donald Friesen. Her email address is lilianag@cs.tamu.edu.