Coordinated Machine Speed Control in a Multiple Filling Machine Installation

Marcus Lindh
Andreas Nilsson

Department of Industrial Electrical Engineering and Automation
Lund Institute of Technology
Lund University, 2006

Coordinated Machine Speed Control in a Multiple Filling Machine Installation

---

Master's Thesis by
Marcus Lindh
Andreas Nilsson

Supervisors
Gustaf Olsson
Dept. of Industrial Electrical Engineering and Automation, Lund Institute of Technology

Johan Nordfeldt
Tetra Pak Carton Ambient AB, Lund

Examiner
Gunnar Lindstedt
Dept. of Industrial Electrical Engineering and Automation, Lund Institute of Technology

Opponents
Christian Isaksson
Andreas Månsson

Department of Industrial Electrical Engineering and Automation
Lund University
Box 118
SE-221 00 LUND
Sweden

http://www.iea.lth.se

Printed in Sweden by Lund Institute of Technology
Lund University
Lund, 2006

Abstract

Multiple installations of filling machines are connected so they take product (e.g. milk) from the same pipe. For a certain type of filling machine this can imply a problem if several machines shape their packages at the same time. When such simultaneous operations happen, large variations in pressure occur in the product pipe, which can lead to decreased package quality. To avoid this, the machines must be non-concurrent.

This thesis investigates what the possibilities are to avoid collisions. It all boils down to affecting the production rates of the machines in a way so they shape their packages at different times. It is shown that there are essentially two different methods to change the production rate. The first one uses a solid-state relay to (during short time intervals) break the current to the main motor in the machines. The second method uses a frequency inverter. Thanks to its lower cost the solid-state relay was chosen as the main alternative, but the possibilities with frequency inverter are also investigated.

The main focus has been to develop an algorithm that performs the coordinated control of the production rates. Extensive simulations and tests have been made in order to find the best solution. We show that it is necessary to allow some collisions if a solid-state relay approach should be used; this to not decrease production rates to unreasonable levels. Furthermore, we show that if no decrease in production rate can be tolerated or if collisions must be avoided to a greater extent, a frequency inverter has to be used.

Preface

This thesis constitutes the fulfilment for our degree of Master of Science in Electrical Engineering at Lund Institute of Technology (LTH). It corresponds to 20 weeks of full time work and was performed in cooperation with Tetra Pak Carton Ambient AB, Lund.

There are a number of people that have been of great help to us and deserve to be acknowledged.

First, we would like to thank our supervisors Johan Nordfeldt, Tetra Pak and Gustaf Olsson, LTH together with the examiner Gunnar Lindstedt, LTH. They have, with their experiences and enthusiasm, had valuable comments and opinions throughout the work. Especially we appreciate the confidence you showed us to develop our ideas and thoughts.

We also would like to thank Johnny Månsson, Thomas Hansson and Bo Norrgren (all Tetra Pak) who assisted us with operating, reconnection and measuring during the machine tests. You all showed great patience.

Finally, we want to thank Karl Paul and Per Holmquist (both Tetra Pak) who contributed with many good advices and viewpoints during our discussions.

Lund
Early spring 2006

Andreas Nilsson
Marcus Lindh

Table of Contents

Chapter 1

Introduction

The milk package is natural and very common on most breakfast tables. In all its modesty it has an important task – to protect its content from outside influence. A good package should also be easy to distribute and handle.

To meet the market's need there are a number of different types of packages. One of these is Tetra Fino Aseptic (TFA), which is a carton based pillow-shaped package, see Figure 1.1. Thanks to its simplicity it brings low costs for both producers and consumers, which makes it attractive to developing countries.

Filling machines are often placed in parallel, where they take product (e.g. milk or juice) from the same product pipe. For TFA/3 (which is the name of the filling machine for Tetra Fino Aseptic) this can imply problems. When several filling machines shape their packages exactly at the same time, large pressure variations can occur in the product pipe. These pressure variations imply decreased package quality. This shows as small deformations, e.g. with folds. Such folds make the package look ugly, but the most serious consequence is that there is a risk it becomes weak and leakage can occur. Another consequence from large variation in pressure is decreased accuracy in package volume.

This thesis explores different possibilities to prevent filling machines from shaping their packages at exactly the same time.

Figure 1.1 The Tetra Fino Aseptic package. (Tetra Pak Carton Ambient AB, 2006)

1.1 Scope

When the filling machine TFA/3 shapes a package a heavy shock occurs. These shocks lead to pressure changes in the product pipe. In the normal case the product pressure is around 0.7-1.2 bar, but if several parallel filling machines takes product from the same product pipe (see Figure 1.2) and, in addition, hits (i.e. when the pressure jaws that shapes the package are

pressed together) at the same time the variations in product pressure becomes very large and can alter between 0-2 bar. In extreme cases underpressure can occur.

Hits from several filling machines that happen at the same time will throughout this thesis be called concurrency. The reason why hits from parallel machines can become concurrent is that they do not keep their exact stated production rate, for example due to friction. To avoid the problems one has to make sure that parallel machines do not hit at the same time. In other words: the machines must *not* be concurrent.

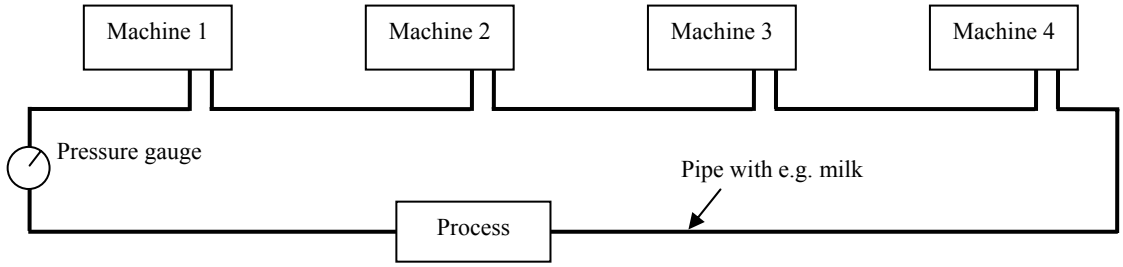


Figure 1.2 Rough outline of a production line with four parallel filling machines.

Something that makes the problem even more complex is that the machines can be of different types. They can produce different package types (e.g. Tetra Brik Aseptic or Tetra Fino Aseptic) and have different package volumes. Machines with different package volumes have different production rates. A TFA/3 that fills 1000 ml or 500 ml packages has the stated production rate 3600 packages/hour whereas a machine that fills 250 ml or 200 ml packages has a rate of 4500 packages/hour. In reality the machines have a certain over-capacity, which gives a production rate of approximately 3770 packages/hour respective 4590 packages/hour.

Another aggravating circumstance is that the number of parallel machines can vary. The solution should be able to handle 2-6 machines. The typical production line consists of 4 machines. Sometimes it can occur that a machine is stopped, either because of maintenance or breakdown. Since the other machines continue their production this is something that must be handled.

1.2 Aim

The aim is to construct a control system which makes sure that multiple installations of filling machines become non-concurrent. Non-concurrent means that they do not hit at the same time, which makes the heaviest shocks in the product pipe disappear. By doing so, fewer packages are deformed and volume accuracy improves.

A demand is that the system should be able to handle up to 6 parallel machines. All combinations of number of machines, package types and volumes are possible, which demands a flexible system. In addition low cost is desirable.

1.3 Approach

For the problems dealt with in this thesis there is hardly any theory. The only part where theory has been available is how to change the production rate of a machine. Everything else in this thesis is founded upon logical reasoning and simulations. The main purpose of the simulations where to develop solution algorithms, but also to get a deeper understanding for the problem. The simulations were made in MATLAB® 6.1 Release 12.1, using Simulink 4.1.

The work behind this thesis can be divided into four main parts.

- Study how the production rate can be changed
- Develop an algorithm that solves the problem
- Implementation of the algorithm in Ladder Diagram
- Verification that the solution works in reality

1.4 Outline of Thesis

This thesis has the following outline:

Chapter 2 Investigation of the different possibilities to change rotation speed on an induction motor.

Chapter 3 Presentation and evaluation of the different algorithms that has been developed as possible solutions to the problem.

Chapter 4 Implementation of the best algorithm, and results from tests on full scale machines.

Chapter 5 Conclusions and suggestions on future potentials

Chapter 2

Fundamental Conditions

It is quite obvious it demands some kind of speed control on the machines in order to prevent them from hitting at the same time. The machines do not necessarily have to be given a different production rate; it may be enough making a machine faster or slower at the precise moment when a collision would occur. This way the machines are given a different production rate in *mean*. Such a selective measure could be performed by braking or just by removing the driving force on the main motor (break the current).

It is also obvious there has to be something that controls *how* and *when* the machines should change their respective rate. An issue to take position over is if the controlling should be made from a separate unit or embedded in a machine PLC. Finally, there has to be some kind of communication between the controlling unit and the machines.

In this chapter the different possibilities are investigated.

2.1 Methods to Change Motor Speed

In order to guarantee that multiple installations of filling machines become non-concurrent, their production rates have to be controlled. The production rate is directly coupled to the rotation speed of the motor shaft, which means that the control task can be performed by changing this rotation speed. The main motor in the filling machines is an induction motor (4 poles) with output power 2.2 kW and 1430 rpm at 50 Hz (2.5 kW and 1720 rpm at 60 Hz). There are essentially three different alternatives to control the rotation speed on an induction motor:

- Physically modify the motor
  - Number of poles
- Change current/voltage
  - Contactor
  - Solid-state relay
  - Frequency inverter
  - Voltage control
- Modify motor load
  - Eddy current brake

The first suggestion can immediately be discarded since it is not possible to change the number of poles. Voltage control is also an unreasonable solution because it is only appropriate on motors with low output power (Herman & Alerich, 1993). The other alternatives are fully viable.

With a contactor or a solid-state relay the rotation speed can be affected by breaking the current during short periods of time. Then the motor is without driving force and it will slow down. If a frequency inverter is used the rotation speed can both be increased and decreased

by changing the frequency to the motor. The eddy current brake applies additional load on the motor which leads to lower rotation speed.

Below is an explanation on every alternative and an appraisal how appropriate it is for this purpose.

2.1.1 Contactor

At present the circuit breakers on the motor are contactors. A contactor is an electromechanical relay designed to break large currents. It consists of an electromagnet and a contact armature (see Figure 2.1). By sending a comparatively small current through the coil of the electromagnet, the contact armature is affected to either open or close the contact. The contactor is electrically isolated between in- and output.

For the purpose of frequently breaking the motor current during short periods of time contactors are inappropriate. The reason is that they are both slow and relatively short-lived. Its slow property arises from the mechanical movement of the contact armature on every switch (on/off). The limited life time is caused by the spark formation that occurs. This spark formation wears down the contact surface and eventually the contact armature cannot make contact. (Bishop, 1986)

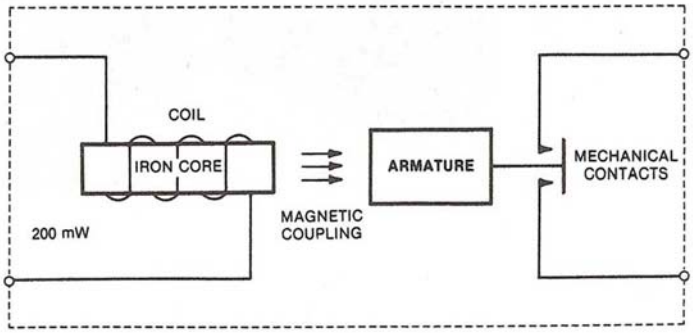


Figure 2.1 Configuration of electromagnetic relay. (Bishop, 1986)

2.1.2 Solid-State Relay

A solid-state relay (SSR) is, just as the contactor, electrically isolated between in- and output. An essential difference is that the coil and contact armature are (in most cases) replaced by a light emitting diode (LED) and a phototransistor (see outline in Figure 2.2). A solid-state relay has no moving parts and is entirely based on technique of semiconductors. Another difference is the way switches are executed. When a contactor receives power it takes a certain time before it makes contact; similarly, when power is removed some time elapses before it brakes. A solid-state relay, on the other hand, switches on when voltage passes zero and switches off when current passes zero. From EMC point of view this is an advantage, but at the same time it brings inaccuracy in *when* the on/off-switches occurs.

SSR is an extremely reliable and long-lived component. For the purpose it is meant to serve here, with many on/off-switches, it is superior to the contactor. This because it is considerably faster and does not have any spark formation like in a mechanical contact, which means that the SSR will not wear out and does not generate any unnecessary disturbances. The disadvantages compared to the contactor are higher heat release and higher price. The higher heat release is quite easily handled with some sort of thermal dissipation.

In the context solid-state relays should be used, there are additional issues to notice. A serious drawback is that SSRs often are closed when they break. This could have far-reaching

consequences, but by using the existing contactors as main switches on the motor the use of SSRs should not bring lower safety. Another detail is that a large current peak can arise in the motor windings if it is turned on when voltage passes zero. This is due to the small impedance in the motor when it is not moving. The current peak is many times larger than the current normally drawn. In this specific case the solid-state relay should not be used to turn on the motor from standstill. Instead the motor will always be rotating and only short disconnections will be carried out. Thus, the current peaks will be suppressed by the electromotive force, but it is still something to consider. Filters to suppress possible voltage or current peaks could be necessary. (Bishop, 1986)

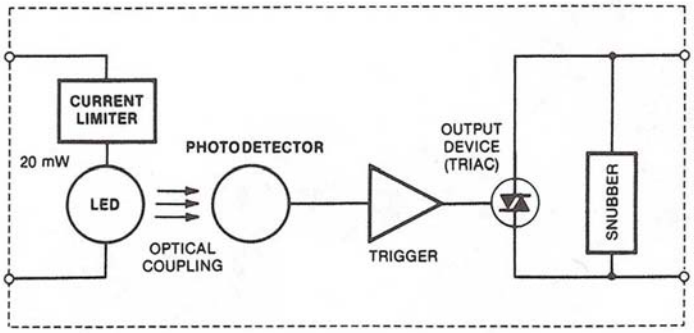


Figure 2.2 Configuration of solid-state relay. (Bishop, 1986)

2.1.3 Frequency Inverter

A frequency inverter can be used to control rotation speed and torque on an alternating current motor. There are different kinds of frequency inverters, but all are built upon the same basic principle. First the frequency inverter converts the alternating voltage to a direct-current voltage by a rectifier. Since the obtained direct-current voltage is not ideal, it must be filtered through a middle circuit. Finally, it is inverted to a new alternating voltage with variable frequency. This new alternating voltage does not have the form of a perfect sine wave. Instead it is constructed from a number of square pulses with different amplitudes and widths. Control circuits ensure the output to have a specific frequency and that the relationship between voltage and frequency is kept constant. This is important since the motor should give the same torque independently of rotation speed. The contents of the four main circuits depend on the type of frequency inverter. (Alfredsson et al., 1986)

By means of a frequency inverter the motor rotation speed can *both* be increased and decreased. This can be done either stepless or in predefined steps. An important issue when using frequency inverters is the use of suitable ramping. This to avoid the additional load a too fast acceleration can lead to.

2.1.4 Eddy Current Brake

When an electrical conducting material moves through a magnetic field currents are induced. These currents interact with the magnetic field and form a force that strives for counteracting the movement. This phenomenon can be used to construct so called eddy current brakes. An eddy current brake consists of an electrical conducting disc together with a magnet. The disc is mounted on the motor shaft so that the magnetic field goes through it (see Figure 2.3). When the motor shaft rotates a current is induced giving rise to a braking force. (Barnes et al., 1993)

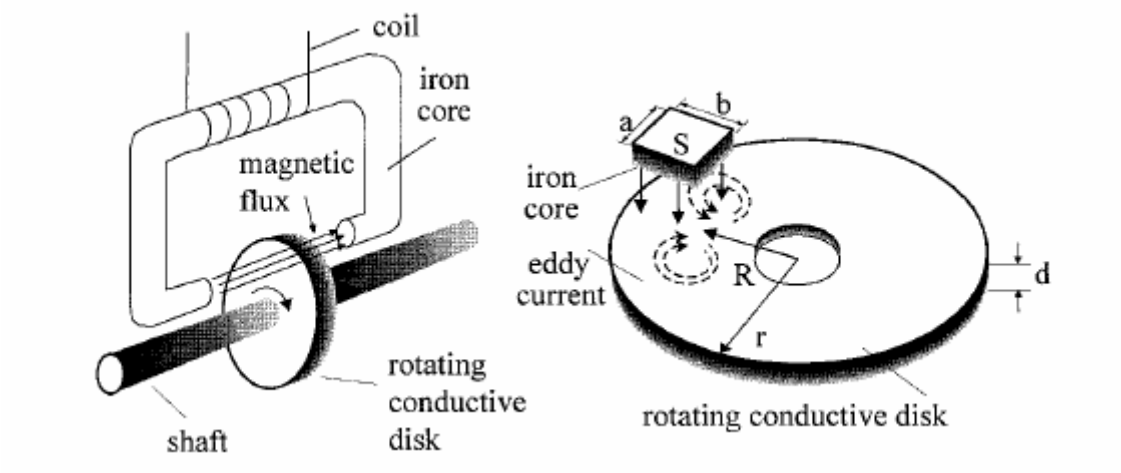


Figure 2.3 Outline of an eddy current brake. (Lee & Park, 1999)

The braking torque for an eddy current brake can be calculated as (see Table 2.1 for parameters):

$$T_b = \sigma R^2 S d \left(\frac{\mu_0 N}{l_g} \right)^2 i^2 \omega \qquad \text{Equation 2.1}$$

Notation	*Description*
T_b	Braking torque [Nm]
σ	Electric conductivity [$\Omega^{-1}m^{-1}$]
R	Distance between centre of disc and centre of magnetic pole [m]
S	Area of magnetic pole [m^2]
d	Disc thickness [m]
N	Number of windings
i	Current through coil [A]
l_g	Air gap [m]
μ_0	Permeability for air [Vs/Am]
ω	Angular velocity of the disc [rps]

Table 2.1 Parameters used in Equation 2.1.

Equation 2.1 shows that the braking torque is directly proportional to the square of the current flowing through the coil. This is no surprise since larger current gives stronger magnetic field. Another thing worth noticing is that the torque is also directly proportional to the angular velocity. This means that the brake action decreases with decreasing velocity. (Lee & Park, 1999)

For dimensioning the brake, the desired braking torque must be known. A reasonable suggestion could be that the motor slows down 0.1 seconds during a 4 seconds time interval. This gives a relative reduction in velocity of 2.5%, which means that the rotation speed of the motor should be 2.5% lower during 4 seconds. As mentioned earlier, the motor has a rotation speed of 1430 rpm at normal working point. During the 4 seconds when braking takes place the rotation speed should be 2.5% lower than 1430 rpm, which gives 1394 rpm. To decide size of the braking torque needed, a typical moment curve of an induction motor can be looked upon, see Figure 2.4 (notice that the curve in the figure does not correspond to the motor in the machine). The curve is almost linear close to the working point. This means that a linearization from two known points can be used to make a good approximation of

necessary braking torque. For both points rotation speed and torque must be known, which implies that the known motor power must be recalculated through Equation 2.2. The obtained torque is 14.7 Nm, which gives the working point (1430 rpm, 14.7 Nm) and the synchronous rotation speed (1500 rpm, 0 Nm) as two known points on the curve.

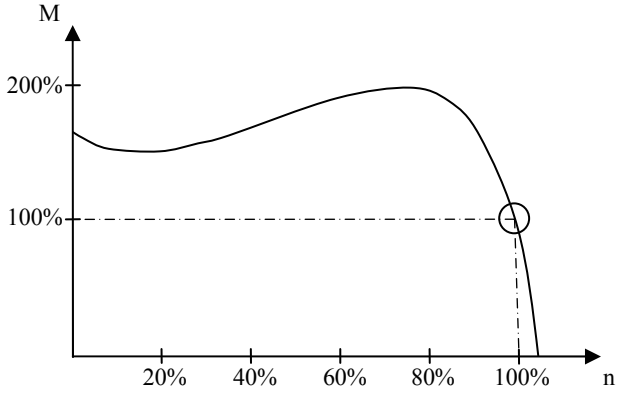


Figure 2.4 Typical moment curve for an induction motor. The working point is indicated by the circle.

$$P = 2\pi\omega T \qquad \text{Equation 2.2}$$

Through linearization it is calculated that the total torque should be 22.21 Nm in order to reduce the rotation speed to 1394 rpm. Therefore the braking torque must be (22.21-14.7) Nm ≈ 7.51 Nm. With reasonable assumptions (see Table 2.2) on the design of the electro magnet the distance between centre of disc and centre of magnetic pole is calculated to 13 cm.

Notation	*Value*
T_b	7.51 Nm
σ	$5.81 \cdot 10^7\ \Omega^{-1}m^{-1}$
S	0.0005 m^2
d	0.01 m
N	2000
i	0.5 A
l_g	0.005 m
μ_0	$4\pi \cdot 10^{-7}$ Vs/Am
ω	23.83 rps (1430 rpm)

Table 2.2 Parameters used when calculating radius of the brake disc.

2.1.5 Comparison between the Methods

Below is a comparison between the suggested methods to reduce the rotation speed of the motor. The aim is to decide the best method.

The contactor is included in the discussion only because it is used at present. As mentioned earlier, it is not designed for making frequent on/off-switches and it has quite long response time. Therefore the contactor is not seen as a possible alternative for reducing the rotation speed of the motor.

A solid-state relay, on the other hand, has properties that make it more suited for the task. Its long life and quickness is something that comes in handy. A disadvantage is the large current peaks that can occur when it switches on. However, this should not give rise to any problems since most modern solid-state relays are equipped with filters.

With a frequency inverter the rotation speed can be tuned to a specified value, which is not possible with the other methods. This means that the necessary speed changes can be done with high accuracy. Also for frequency inverters, filters have to be used to take care of disturbances. A big advantage is that the frequency inverter not only can decrease rotation speed but also increase it.

The introduction of eddy current brake demands large reconstruction of the machine. To realize the suggestion a metal disc has to be mounted on the motor shaft. In addition, an electro magnet must be placed on the outer edge of the disc. According to the calculations the diameter of the brake disc must be at least 30 cm. Together with the space needed by the electro magnet the total space needed would be around 50 cm in diameter. Another disadvantage with the eddy current brake is the power dissipation it causes.

From this reasoning it is established that solid-state relay and frequency inverter are preferred. Solid-state relays have the cheaper price but frequency inverters provide the best solution. After discussions with Tetra Pak, it was determined that solid-state relay should be the main alternative, but the opportunities with frequency inverter should also be investigated.

2.2 Input and Output Signals

To make the filling machines non-concurrent some kind of two-way communication is needed. Since it is communication between two PLCs, digital signals are preferred. In order to fulfil the control task the following signals are required. Every machine has to output two signals; one should indicate when a machine hits (in this section called “hit”) and the other should indicate that the machine is in production (“running”). Machine in production ought to be a prerequisite for performing any control. The first signal should be true only when a machine hits and the second should be true all the time when the machine is in production. If solid-state relays are used, one input signal is needed on every machine. The signal should be sent from the control unit and be an indicator when the machine should make itself slower. If a frequency inverter is used two input signals are needed, one for increasing and one for decreasing the frequency.

The two output signals, mentioned above, have to be constructed within the PLCs on the machines. The signal “running” can quite easily be constructed from two Boolean variables that already exist. For construction of “hit” an angle encoder can be used. The angle encoder indicates were in its production cycle the machine is. The cycle can be thought of as a circle. During one cycle two hits occur, divided by 180°. If it is known where (on the cycle) the hits occur, this can be used to construct the signal “hit”. The angle encoder can represent numbers between 0-255 (8 bits), which gives a resolution of 1.41°/bit.

Another issue to decide on is whether the machines should handle the control themselves, or if an external PLC should be used. If the control system is internal it must be implemented in every machine in the production line, because it must work even if a machine is out of production. Since the PLCs in the machines do not support network communication all signals have to be sent through separate cables to and from all the machines, which leads to extensive wiring. Installing a PLC with network possibilities would be the best solution, but due to the significantly higher cost this is not in question. Therefore, an external PLC will be used.

Chapter 3

Analysis of Possible Algorithms for the Machine Speed Control

In the previous chapter it was discussed which different possibilities are available when the decision already has been made to change the speed of a machine. What decides *when* a machine speed should be changed is investigated in this chapter.

The chapter begins with several different algorithms which, during their development, all were seen as possible alternatives to handle the control. The design of the algorithms is dependent on whether they use solid-state relay or frequency inverter. For each algorithm its principle is first explained, followed by results and conclusions from simulations. The simulations were made in Simulink and have been an absolutely necessary aid to evaluate the eligibility of the algorithms. Through mental effort one can realize what happens when two parallel machines are running, and to some extent also three, but after that it is almost impossible. This is where the greatness of simulations is shown, since it allows thorough analysis why some situations can occur and what could be done to avoid them. The algorithms are explained in the order they were developed.

From now on several terms will be used. One of these terms is hit interval, which refers to the smallest time difference allowed between two hits of any machines before they are considered to collide, see Figure 3.1. A reasonable hit interval should be in the range 50-100 ms; the simulations will show what hit interval to use. A small hit interval implies that two hits are allowed to be close to each other, and vice versa. When the time difference between hits from two machines is close to the hit interval a delay (for most algorithms) should be made. A delay is meant to increase the time difference between hits from two machines. Another term that often will be used is base speed, which is defined as the lowest guaranteed speed a machine is allowed to have. Within each base speed the period (time difference between the machine's hits) of the machines only differ a few thousandths of a second. The algorithms treated in this chapter can handle machines with two different base speeds, 3600 and 4500 packages/hour. Block diagram for each algorithm explained in the chapter can be viewed in Appendix A.

3.1 Solid-State Relay

In this section the algorithms developed for use with solid-state relay are described. The quality of an algorithm is, besides the avoidance of all hits, dependent on the change in production rate it causes. With a solid-state relay the machines can only become slower, which means that less interference results in higher production rate.

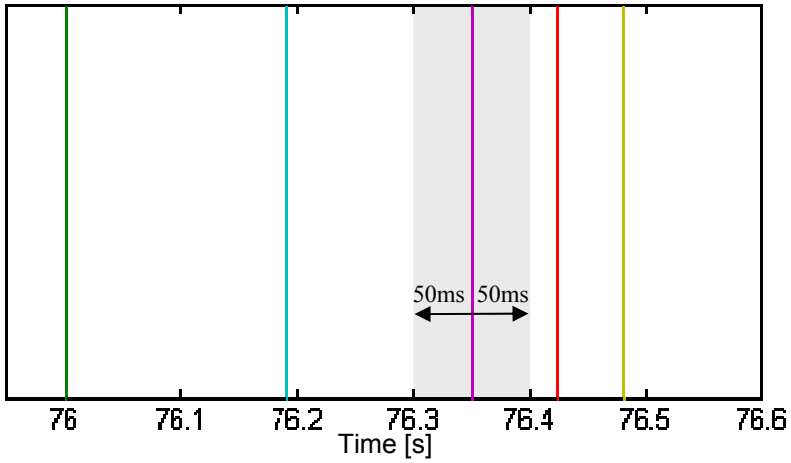


Figure 3.1 Hits as they were represented during simulations. Each vertical line represents a hit. Hits from different machines are represented with different colours. A hit interval of 50 ms for the purple machine is marked with the arrows; in this case there are no other machines within the hit interval and therefore no collision.

3.1.1 Algorithm I – Equal distribution

At first sight of the problem a good and simple solution seems to be to distribute the machines' hits so the time difference between them is as large as possible. A large time difference implies that the influence between them is minimal. In order to maintain the largest time difference between hits, the average speeds of the machines have to be equal. Corrections must be made continuously to make the speed of each machine relatively close to the average speed. Naturally this reasoning can only be applied when machines have equal base speed.

The algorithm *Equal distribution* first determines which of the machines are in production. After that, the number of machines in every base speed is calculated. The machine with the lowest production rate in every base speed is appointed to master. The master machine is allowed to maintain its production rate, while the other machines are delayed to make their average production rate become equal to the production rate of the master. The desired time between the hits from two machines is calculated as the quotient between the master's period and the total number of machines within the base speed, which means that the hits are equally distributed over the master's period. Hence, the time difference between two hits decreases when number of machines in production increases. The result when hits from all machines are equally distributed is shown in Figure 3.2.

The hits from the slowest machine (master) are placed leftmost in every period. If all machines started at the same time they would be ordered, from left to right, with descending period time. However, if a machine is started after the others, it will stay where it first was placed. If the latest machine is slower than the current master, *it* will become the new master which the other machines would adjust their speed to. In case the new machine is faster than the master it will remain in the position where it first was placed, even though it has a slower machine to the right. The reason to this is that the delays used for this algorithm are not long enough to let a machine pass another machine.

Every time a machine hits, the algorithm calculates how the machines' hits are placed one period ahead. When a collision is predicted it is only the machine that recently hit which can be delayed (due to mechanical reasons, further discussed in Section 3.3). The prediction of a

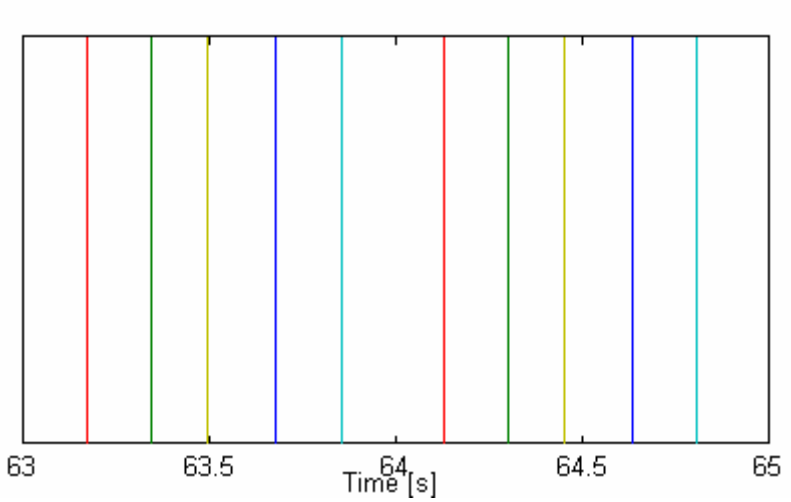


Figure 3.2 Distribution of the hits from 5 machines when *Equal distribution* is used. Two periods are shown and the red machine is the slowest (master).

collision does not automatically imply that the machine should be delayed; it can be the other machine in the collision that should be delayed.

The basic principle to avoid a collision is to delay the fastest machine, but if the fastest machine has come too close to the master from the left (i.e. it has been delayed so its distance to the master is too small) it is the master that should be delayed. The reason to this is that the faster machine otherwise would collide with the master and then get the wrong position. In case the production rates are equal, the machine that is positioned to the right in the collision is delayed.

When it is determined that a machine should be delayed, the algorithm investigates if a delay results in a collision between the machine and another machine with different base speed. If this is true the delay will not be made. The main reason for this is that *Equal distribution* considers two machines to collide when the time difference between two machines' hits are less than the quotient. Since the quotient is larger than the hit interval, the delay can be made next period instead. Still, the result will not be a collision. Hits from the machines with same production rate will only get closer to each other. A delay will be made if all tests are passed and the conclusion is to make a delay.

Results from simulations for *Equal distribution* shows that the algorithm can avoid all collisions after the transient state has passed. This is naturally a fundamental prerequisite if the algorithm is going to be used. Performance of the algorithm is entirely dependent on the length of the delays together with a slight modification of the quotient (see below). Too long delay implies that the machine with the last hit in a period would collide with the first hit in the next period (master). However, if the delay is too small this would mean that a faster machine cannot become slow enough to remain its distance to the master. Simulations show that a suitable length of delay is 50 ms.

If the quotient between the master's period and the number of machines is used without any modification, the algorithm requires the machines' hits to have exactly the correct distance from each other all the time. This would imply that the distance between the last hit in a period and the master's hit in next period is the smallest possible. A delay on the last machine would position it too close to the master, whereupon the master has to delay. This would start a train where all machines have to delay. Such behaviour is naturally undesired and simulations show that the number of delays becomes around three times more. To avoid this behaviour the quotient has to be made approximately 5 ms smaller. The result will be closer hits, but the margin to the master in next period will be larger. By comparing Figure

3.2and Figure 3.3 the effect can be visualized. It is Figure 3.2 that corresponds to a 5 ms decrease of the quotient.

Even when the suggested values above are used, the algorithm needs a lot of delays to maintain the desired appearance shown in Figure 3.2.

Tests shows that the large number of delays would (for at least some machine) decrease production rate 1-2%. This is a fairly substantial deterioration of production rate. It is easy to believe that this would put some machine below the lowest required production rate. This is however not the case, since the algorithm always delays the fastest machines in each base speed. In other words, the machines are guaranteed not to get below the required production rate. Even so, the large number of delays is the reason why *Equal distribution* is fairly uninteresting; there must be another smarter way to avoid collisions.

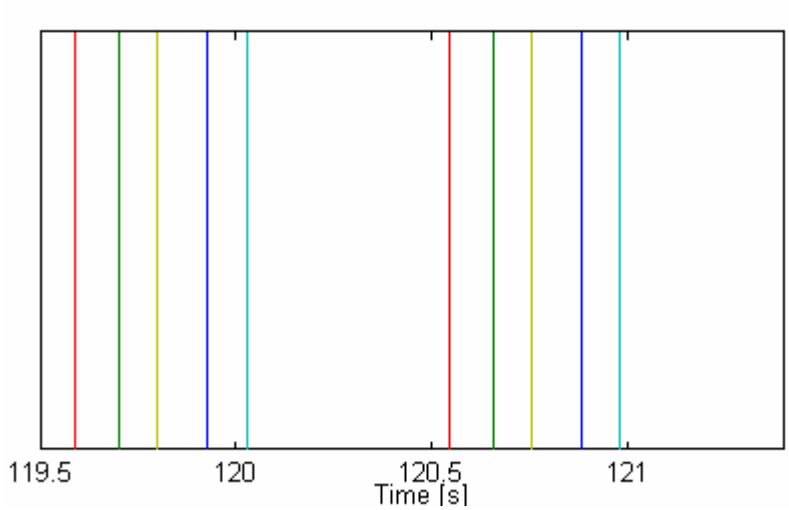


Figure 3.3 The hits from 5 machines placed closer together than Figure 3.2. The red machine is the slowest (master).

3.1.2 Algorithm II – Fox jump

A way to avoid the large number of delays caused by *Equal distribution* is to let every machine retain its production rate until it gets too close to a hit from another machine. When the hits for two machines with same base speed are too close, the slower machine will be delayed to let the faster machine pass without a collision, see Figure 3.4. The algorithm relies on that the delay can be made as long as two hit intervals. This is very important, since a machine otherwise cannot "jump over" another machine without collision. The jumping behaviour is a partial reason to the name *Fox jump*.

A difference from *Equal distribution* is that *Fox jump* does not need any knowledge of the number of machines in production for each base speed. Neither does it need to calculate which machine that got the lowest production rate. The algorithm *Fox jump* waits for any machine to hit. When a hit has occurred the algorithm calculates how the machines' hits are positioned one period ahead. Even if a collision is predicted it is not certain that the machine should be delayed. First the algorithm checks whether a delay causes a collision between two machines with different base speed. If that is the case the machine is not delayed. The reason is that the other machine involved in the collision can be a slower machine, which has already had its hit and therefore is unable to avoid the collision. There is no reason to avoid one collision by creating a new one.

Simulations for *Fox jump* shows that the size of the used delay is crucial for how well the algorithm can solve its task. As mentioned before the delay has to be at least twice the hit interval. Otherwise the machines will collide even though one is delayed. However, with a too

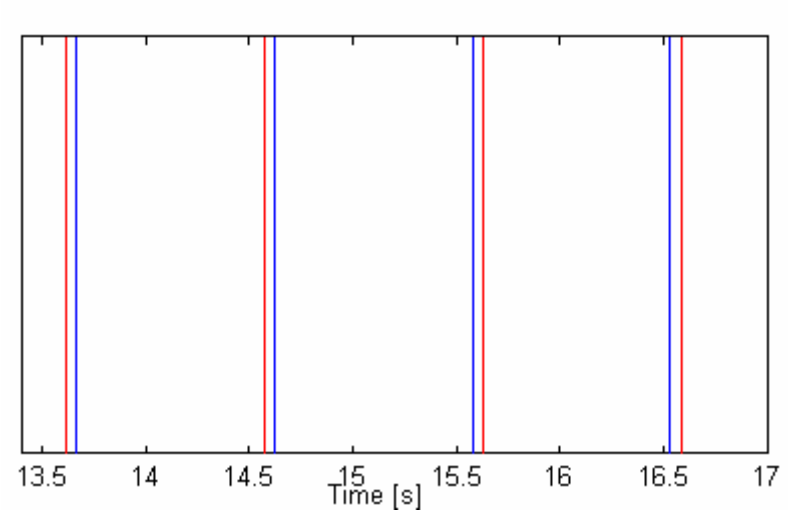


Figure 3.4 Visualization of a so called fox jump. The blue machine is slightly faster than the red. When the distance between them is predicted to be within a hit interval the red machine is given a delay so that the blue can pass.

long delay it happens quite often that the delayed machine will collide with another machine. This would start a domino-effect, which results in a huge number of delays. According to the simulations a delay of 100 ms would be the best trade-off between decreased production capacity and separation of the machines' hits. This delay implies a minimum distance of 50 ms between the hits of two machines, before it is considered to be a collision.

The results of simulations show that the algorithm *Fox jump* only requires less than half the delays needed by *Equal distribution* to avoid collisions. Thus, the change in production rate is considerably smaller.

The fact that it is the slowest machine which is delayed every time two machines collide is naturally a disadvantage. This means there is no guarantee for the lowest production rate. The algorithm is however saved by the machines' over-capacity in production rate.

3.1.3 Algorithm III – Always delay slow

As mentioned before the solution should not be restricted to only handle machines with the same base speed. Even a mixture with different speeds should be allowed.

Since the faster machines are getting nearer the slower, it is natural to come up with a solution where the slower machine lets the faster pass. This can be done if the slower machine is delayed every time a collision between machines of different base speed is predicted. The algorithm requires that it is possible to delay a machine twice the length of a hit interval.

The algorithm *Always delay slow* waits for a machine to hit. At that moment it is calculated whether the machine that has hit will collide with a machine with different base speed. If this is true the machine that has hit will be delayed. There is no need to let the algorithm decide which machine that is the slowest in the collision. The machine that the collision was calculated from naturally has the longest period and is consequently the slowest machine.

Simulations show that the algorithm can avoid collisions. However, it depends on the delay to be twice the hit interval. At the same time the delay cannot be too long, by same reason as *Fox jump*. Even though *Always delay slow* can solve its task, it is not an interesting algorithm. The reason is, as the name says, that it always delays the slowest machine, which leads to a lot of delays for that machine. Simulations show that the slowest machines sometimes lower their production rate by more than 5%. Normally this means that the delayed machine would get below the required production rate.

3.1.4 Algorithm IV – Logical delay

To avoid the large number of delays some machines were subjected to when *Always delay slow* was used, it would be desired to also delay the faster machine in a potential collision. This would decrease the number of delays for the slowest machine, which increases its production rate. An algorithm with this purpose is *Logical delay*. The fundamental principle of this algorithm is to also delay machines with the faster base speed, and in that way distribute the delays more evenly between the machines.

Like the previous algorithms *Logical delay* waits for a machine to hit. At that moment it calculates if the machine will collide with a machine of different base speed. When a collision is predicted, the algorithm checks how the hits are positioned in relation to each other. If the faster machine is predicted to hit before the slower, a collision will be avoided by delaying the slower machine. On the other hand, if the faster machine will hit after the slower, the algorithm will choose to delay the faster machine. The reason to this decision is that the faster machine will pass the slower by its own efforts in next hit. The smallest delay possible to use is equal to the length of the hit interval. The reason is that one of the machines must be delayed the entire hit interval if both machines hit at exactly the same time. This is a difference compared to the algorithm *Always delay slow,* where the delay must be twice the hit interval. It is not eligible to choose a too long delay since it in the end only leads to decreased production rate.

As expected, simulations for the algorithm show that the delays are more evenly distributed between the machines. The slowest machine in the system is not delayed as often as for the algorithm *Always delay slow*. In a simulation with five faster machines and one slower (same simulation as for *Always delay slow*) the production rate of the slowest machine is decreased by about 3%. This is an improvement and the algorithm is therefore preferred ahead of *Always delay slow,* but still it is a large decrease in productivity.

3.1.5 Algorithm V – Allow 2-hits

This far the algorithms have caused substantial negative effect (possible exception is *Fox jump*) on the production rates. A way to elude this is to lower the demand of avoiding all collisions. The algorithm *Allow 2-hits* allows collisions that occur between two machines' hits, but not collisions where more machines are involved.

The algorithm waits for a machine to hit. At that moment it calculates if the machine will collide with another machine in its next hit. If a collision is predicted it is investigated if there is a third machine involved. If that is the case it must be calculated which machine has the rightmost hit (last hit) in the collision, see Figure 3.5. It is the machine with the last hit which should be delayed, but this is not always feasible. If the last machine already has been delayed it can not get a further delay. In that case, the algorithm investigates if the machine in the middle can be delayed. If also this is impossible the collision between three machines can not be avoided (unless the leftmost machine is given a very long delay).

A possible scenario is that two machines with exactly the same production rate will collide with each other. According to the algorithm's basic principle this would be allowed to happen without any correction being made, which is not desired. Since there is no difference in production rate between the machines they will collide with each other until a third machine is involved and the algorithm takes care of that collision. To avoid this, there is a separate part

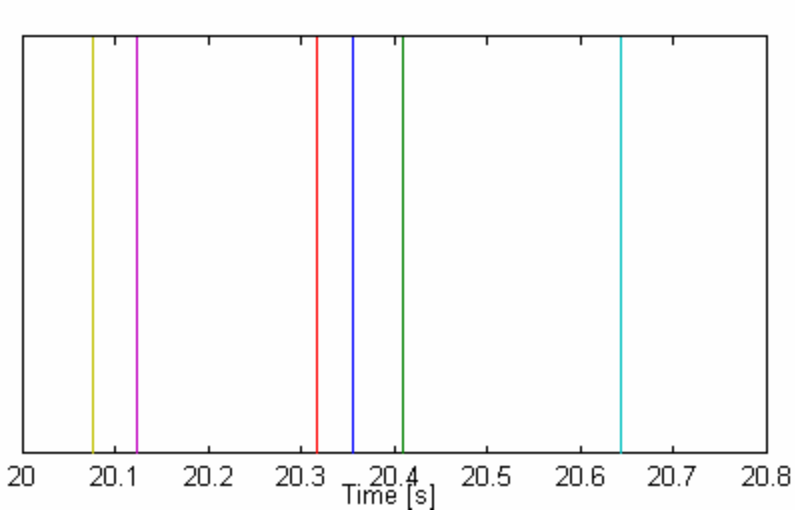


Figure 3.5 Visualization of a collision between three machines (red, blue and green). The green machine is said to be the "last" and blue to be the "middle" in the collision.

which takes care of collisions between machines with exactly the same production rate. The algorithm decides to delay the machine positioned last in collision.

Simulations show that the algorithm does as intended. It allows 2-hits while collisions involving more machines are avoided. The number of delays required is of course considerably fewer compared to other algorithms. As for *Logical delay* the delay needs to be at least the length of the hit interval. Like for most other algorithms, simulations show that a long delay not only decreases production rate, it also increases the risk to delay a machine into further collision.

3.2 Frequency Inverter

The main difference between solid-state relay and frequency inverter is that the latter not only can make a machine slower but also faster. Besides, a frequency inverter can really change the speed of a machine. This is a difference compared to using solid-state relay, which needs to delay a machine occasionally to change its average production rate. The properties of a frequency inverter open new possibilities to avoid collision between machines.

It should be pointed out that the following algorithm is not fully developed.

3.2.1 Algorithm VI – Equal distribution (frequency inverter)

According to the reasoning when developing algorithms for solid-state relay, the most advantageous solution should be to equally distribute the machines' hits. Consequently, the time distance between them would be as long as possible, see Figure 3.6. The hits would then affect each other as little as possible. To accomplish this all machines must have exactly the same average production rate. Since a frequency inverter can make a machine faster, all machines within a base speed can obtain the rate of the fastest machine (master). This is a significant difference compared to using solid-state relay, where all machines are delayed to change the production rate and therefore decreases the total production. A frequency inverter can instead increase the machines' production.

The algorithm *Equal distribution (frequency inverter)* first calculates the number of machines in production for each base speed. For every base speed the fastest machine is appointed to master. The other machines should increase their speed to get the same production rate as the master. As before, the desired time difference between two machines' hits is calculated as the quotient between the master's period and the number of machines within a base speed. The control of the hits is accomplished by cascade-connection of two proportional controllers, see Figure 3.7. The inner controller in the cascade-connection controls the speed of a machine so it obtains the same speed as the master. The outer controller adjusts the time distance between the machines' hits within same base speed to be as long as possible. The algorithm calculates every tenth of a second (sample rate) if the speed of the machines should be increased, decreased or unchanged. First the algorithm predicts when the next hit for a machine will occur. If this hit will be positioned too close to a hit from another machine with same base speed, the outer controller will get an error that must be adjusted. The machine either increases or decreases its speed in order to position itself correctly. When the time difference is correct the algorithm investigates what speed the machine has compared to the master. The inner controller controls until the speeds are equal.

The algorithm is written in a way that does not compare hits from machines with different base speeds. Hence, collisions between two machines are allowed. When the machines' hits are equally distributed the algorithm maintain this appearance until a new machine is appointed to master, e.g. if the fastest machine is taken out of production. The algorithm must then control the hits of remaining machines and distribute them equally with the new production rate.

The program sends an output to the machine as long as its speed should increase or decrease. The frequency inverter then accelerates the speed like a ramp, which in the simulations was set to 2 Hz/s. The algorithm continuously calculates which speed the machines have internally. This is necessary since new information about the real period for a machine is only available when it hits. There is also a limitation in the program to prevent the machines to get below required production rate.

Simulations show that the algorithm works well and distributes the machines' hits equally for the different base speeds (see Figure 3.8). Since the two base speeds are treated independently, collisions between two machines occur. Collisions between more than two machines are impossible with only two different base speeds. The advantage to allow collision between two machines is that no adjustment is needed once the machines' hits are equally distributed. This is a considerable difference from using solid-state relay which frequently has to break the current to the motors to change production rate. From simulations it can also be shown that production increases for all machines except the master in every base speed.

3.3 Conclusions from Simulations

The algorithms developed for solid-state relay predicts *one* hit ahead if a collision between several machines' hits will occur. This is necessary to be able to discover a collision before it has happened and consequently delay one of the machines involved to avoid the collision. However, there are occasional scenarios where collisions can not be avoided by the algorithms. The most common reason for this is that machines involved in the collision already have been delayed to prevent another collision. This implies that longer prediction would be desired, but because of the considerable increase in complexity it is not implemented in any of the algorithms.

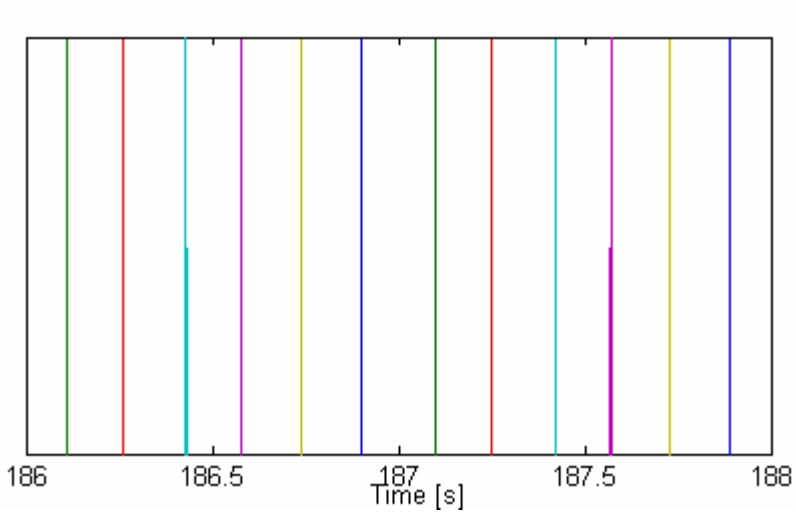


Figure 3.6 Distribution of the hits when frequency inverter is used.

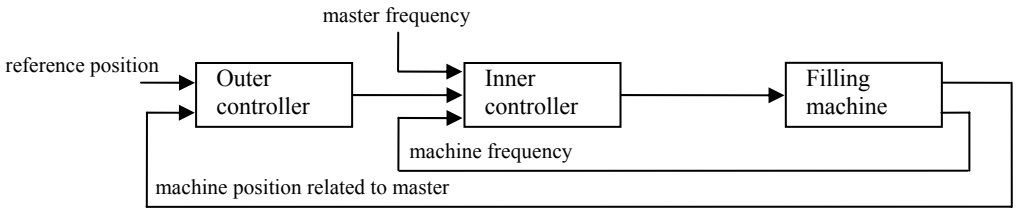


Figure 3.7 Rough outline of cascade controller.

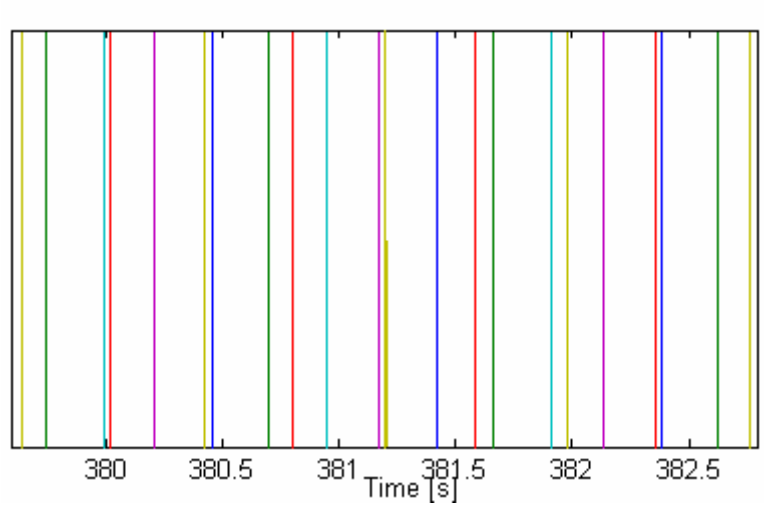


Figure 3.8 Frequency inverter used for two different base speeds. The hits within a base speed are equally distributed. (Green, cyan, purple and blue are slower machines – red and yellow are faster.)

If an algorithm uses prediction more than one hit ahead it means that a machine can be delayed much earlier before the collision occurs. However, it is here the complexity shows. A delayed machine can in worst case collide with other machines before the original collision occurs. In this way new collisions, that were not originally there, have been created. In order to avoid creating new collisions the calculations would be too heavy. Simulations also show that the collisions that could be avoided this way are very rare, and never involve more than two machines (three for *Allow 2-hits*) and are almost always positioned in the outer edge of the hit interval. With this knowledge and that the algorithm should be able to run in real-time with a sample rate of 1 ms, prediction only one hit ahead is to prefer.

The algorithms for solid-state relay can only make machines slower, which is the main difference compared to the algorithm used for frequency inverter. Of course, this limits the possibilities to avoid some collisions. Another limitation, which is not quite obvious, is that the algorithms are not allowed to delay other machines except for the one that just hit. The reason is that a certain time is required to make a machine slower. The time required corresponds to the time the current to motor has to be broken to attain a certain delay effect. Since the machine is somewhere between two hits, it may be too short time to next hit to be able to make a delay of desired length.

For all algorithms the length of the hit interval has great effect on the number of delays. To minimize the interference between two machines' hits, the hit interval should be as long as possible. However, a larger hit interval results in longer and possibly more delays which decreases the production rate. Simulations have shown that it is not suitable to use a delay longer than 100 ms. This implies a maximum hit interval of 50 ms for the algorithms *Fox jump* and *Logical delay* and 100 ms for the others.

Furthermore, simulations show that even if the delay is 100 ms, it will cause substantial negative effect on the production rate. The algorithm *Allow 2-hits* is therefore developed as a compromise to avoid hits and at the same time not lower the production rate too much. However, there are still some collisions between two machines that are desired to avoid. This is when two machines with same base speed get too close to each other. Since the machines' production rates are almost equal, they would collide during a long time if no measures are taken. The most eligible algorithm to solve this is *Fox jump*. It separates the hits if they get closer than 50 ms to each other. At the same time the algorithm does not lower the production rate too much. Thus, the algorithms *Fox jump* and *Allow 2-hits* is the best combination to meet the demands.

Chapter 4

Experiments and Implementation on the Full Scale Machines

This chapter treats the different aspects when it comes to using the selected algorithm on the full scale machine.

4.1 Investigation of Braking Effect in the Machine

The aim of this test was to investigate for how long the current must be broken to the main motor in order to extend the period with 100 ms. Since the braking effect is dependent on the inertia in the machine, it is of most importance that it is performed at an appropriate moment in the production cycle. Where this moment is was also investigated.

The test was performed on a machine that manufactures approximately 3600 packages/hour (500ml).

4.1.1 Preparations

To discover where in the production cycle the pressure jaws hits and where the heaviest sectors are, the machine was cranked by hand-power. By watching the machine and at the same time read the angle encoder, the desired values were obtained, see Table 4.1. It is the decimal numbers in the table that are used in the Ladder-program. An interesting observation was that one could feel how the machine was significantly heavier to crank in the heavy sectors of the production cycle.

Description	*Octal number*	*Decimal number*	*Decimal degrees*
Hit Left	50	40	56.5
Hit Right	250	168	236.5
Delay sector Left (start)	140	96	135.5
Delay sector Right (start)	340	224	315.5

Table 4.1 Angle encoder-values for hits and heavy sectors. The heavy sectors are approximately 65˚ long.

4.1.2 Approach

To be able to do the test some wiring had to be done in the machine. A self-holding contactor was strapped and the brake on the main motor was reconnected so it did not brake automatically when the current to the motor was switched off.

Three sweeps were made with break lengths of: 350-550 ms, 400-600 ms and 200-400 ms. Since every sweep had 5 ms steps, it gave 40 delayed periods. In addition, a reference

measurement was made where the machine ran without any delays. To make sure two delays did not affect each other, there were three normal periods between the delays.

The following signals were measured:

- one phase to the motor (line voltage)
- PLC output that controls the line voltage contactor
- PLC output that controls the sealing pulse (TS pulse)
- PLC output indicating when the machine hit

The sampling period was 1 ms for all the signals. The main reason for this is that the solution algorithm will use that sampling interval.

4.1.3 Results

When the testing began it was discovered that the periods of the machine were not completely symmetrical (see Figure 4.1). The difference, which was 3 ms (0.952 s against 0.955 s), could be caused by mechanical differences between the two halves of the production cycle or by the resolution in either PLC or measurement equipment.

In Figure 4.2, it can be seen how the line voltage is affected after that the PLC has given signal to the contactor. The time delay between the PLC output to the contactor and the drop in line voltage originates from properties of the contactor. The relay has a switch off delay of approximately 35 ms and a switch on delay of about 45 ms. The difference of 10 ms makes the drop in line voltage 10 ms longer than the PLC signals. The delay at switch off implies that the delays are not made exactly where they were intended. Testing with contactors instead of solid-state relays still has the advantage of being fully predictable. This is a difference to solid-state relays, which switch off when current passes zero and therefore have an uncertainty of half a period (10 ms at 50 Hz). Furthermore, the figure shows how the break is made in the middle of two hits and stops before next the hit, which is very important since next period should not be affected.

In Figure 4.3 period times from a sweep with current breaks between 350-550 ms are shown. The high peaks correspond to periods where the delays are made. Between the delayed periods there are three periods without a delay, as mentioned before. On the peaks of the delayed periods a wave form can be perceived. We have no good explanation to this and since it has no influence on the test it is left without further investigation.

Without any delay the period is 0.955 s, and with breaks of 350-550 ms the period increases to 1.020-1.080 s. The calculated differences are shown in Figure 4.4. The grey curve relates it to the break length desired from the PLC and the black curve relates it to the real length of the current break. It can be determined that a current break of approximately 475 ms is required to make the period 100 ms longer. The exact value is presumably individual for every machine.

When the curves level off, it is because two different break-times from the PLC have resulted in approximately the same delay. It also occurs that the curve becomes almost vertical, which means that the same break-time has resulted in two different delays. This may seem strange since a sweep did not do any delay more than once. The explanation is that it is not easy to determine exactly for how long the line voltage has been broken. When constructing the graph we defined the current break as the time from the first amplitude below 300 V to the first amplitude above 270 V. For specific breaks this can give an uncertainty of around 5 to 10 ms, but for the entire sweep it gives a good picture on the relation between the real current break, the output from the PLC and the resulting delay.

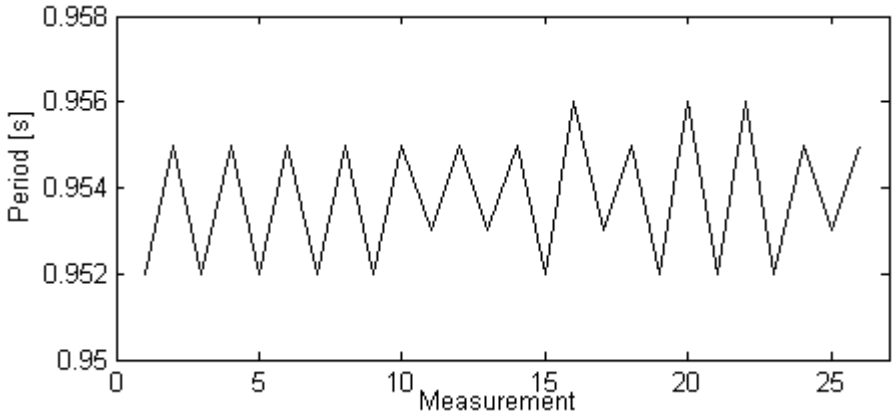


Figure 4.1 The asymmetric behaviour of the period time.

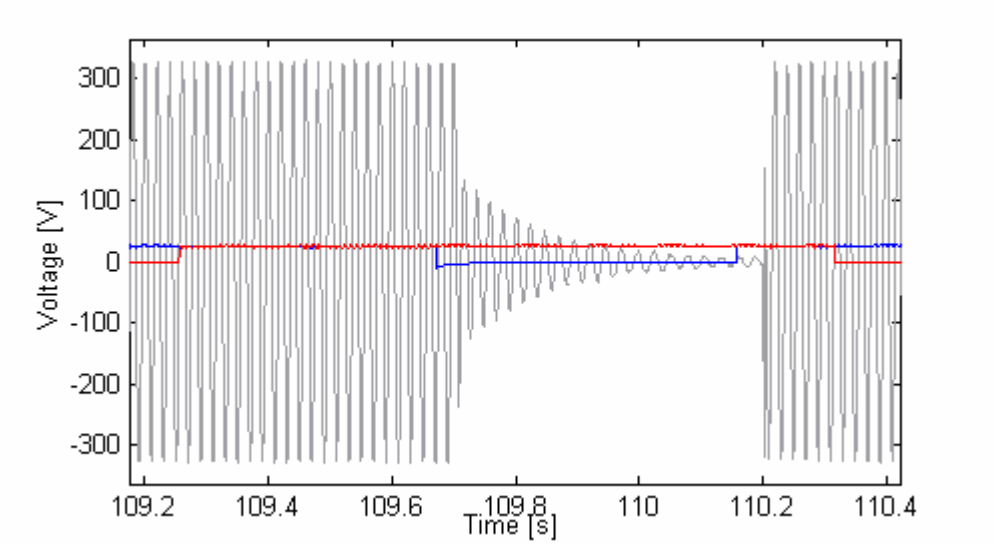


Figure 4.2 The line voltage (grey) when the PLC gives break-signal to the contactor (blue). The machine's hits are indicated by transitions (red).

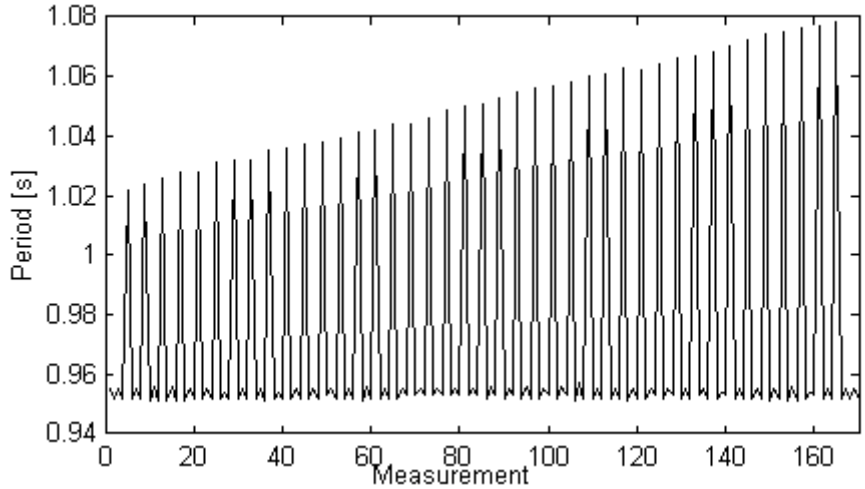


Figure 4.3 Period times from a sweep with current breaks to the main motor between 350-550 ms. Every edge in the graph represents one period time.

To see where the limit is before a delay affects the following period, a sweep between 400-600 ms were made, see Figure 4.5. For the last delays in the measurement (around 550-600 ms) it is clear how it also affects the length of the following period. This is however only an observation of where the limit is and does not make any problem, since we are not interested in breaking the current that long time.

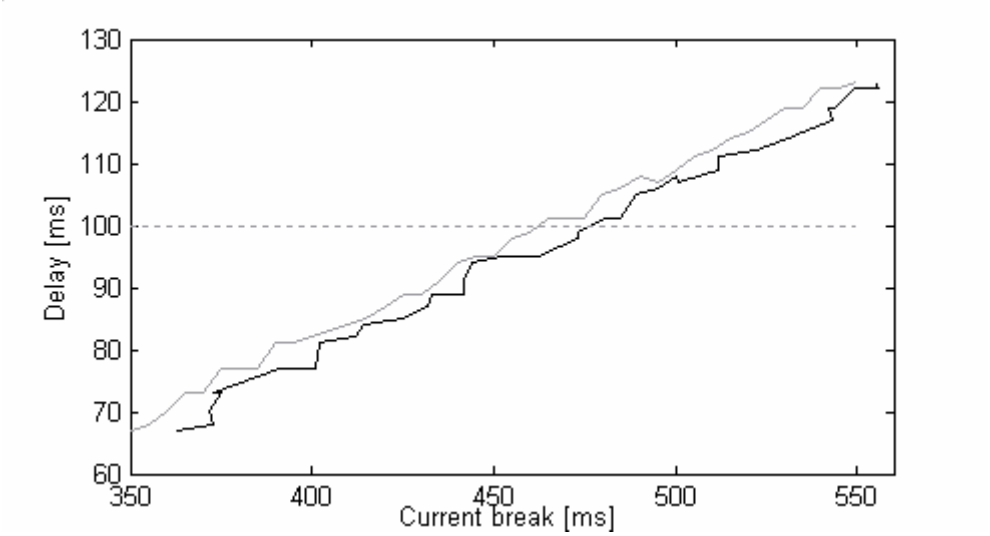


Figure 4.4 Length of delay related to length of current break to the main motor (black) and desired current break from PLC output (grey).

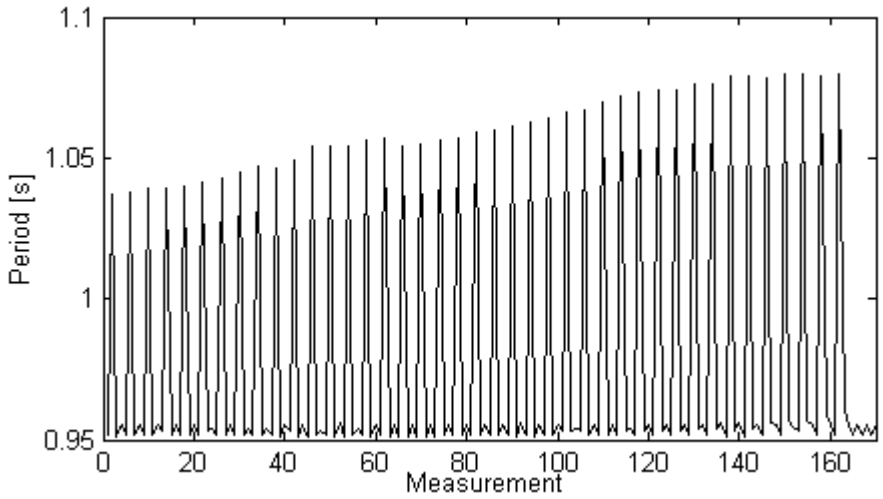


Figure 4.5 Period times from a sweep with current breaks to the main motor between 400-600 ms. Every edge in the graph represents one period time. During the last 6 delayed periods also the subsequent periods are affected.

4.2 Implementation

This section describes different aspects when it comes to implementing the selected solution algorithm.

A choice made earlier is that the control should be performed from an external PLC. Its task is to determine whether a machine should delay itself. The decisions are communicated through a logical signal directly to the PLC in respective machine. The machine waits until it reaches the heaviest production sectors before the delay is carried out.

As mentioned earlier, all implementation in the PLCs must be done in the programming language Ladder Diagram. For the most part Ladder consists of graphical symbols representing logical expression. Additionally, there are a number of more complex functions, which for example can handle numbers.

The program is built with different blocks. For example, the two algorithms (*Algorithm II – Fox jump* and *Algorithm V – Allow 2-hits*) that make the solution are placed in two separate blocks. This makes the implementation easier to read and it offers an opportunity to easily turn the different algorithms on or off. The blocks mentioned this far are executed once every program scan.

A considerable difference between Simulink and the PLC is that Simulink guarantees that every line of the program is executed on every sample. This is possible because Simulink uses simulated real time. When the entire program code has been executed the simulation jumps to the next sample and runs the code there, with the conditions *that* sample brings. The PLC can of course not give this kind of guarantee. Instead the program lines that must be executed every sample can be placed in a special block (called fastscan), which runs according to timed interrupts. In fastscan there are instructions that either must be executed at every sample, or must know with what interval it is executed (in this case a time counter). The block fastscan can be executed many times during one program scan. Consequently, it interrupts the normal program at every new sample. The result of this can be that variables that are given their value in fastscan and used in the normal program can change during a program scan. This can of course have catastrophic consequences for the results of the algorithms. To avoid this problem the affected variables are copied in the beginning of every new program scan.

Another issue to consider is to be restrictive with what to place in fastscan. Since it is an interrupt block every new line makes the execution time significantly longer. In an extreme case fastscan can take such a long time to execute that next interrupt is missed. In our program the signal “hit” from the machines are placed in fastscan together with calculation of the machine periods.

The outputs from the control unit are indications to respective machine that it should make a delay. When the central unit has decided on a delay the corresponding output is turned on, which is done during the same program scan as the hit is registered. Since there is a small time gap between the hit and the heavy sector in the machine, the outputs do not need to be immediate.

The simulations in Simulink gave not only understanding for the problem and possible solutions. It also gave a favourable understanding for the PLC implementation. This because almost everything in the simulations were based on written program lines. From there it was quite straightforward to translate into Ladder-code. However, some problems were encountered when using arrays but these could be solved with help from special function blocks.

Also the program lines in the machines were quite straightforward. The angle encoder, mentioned earlier, is used to determine both when a hit occurs and when the machine enters its heavy sectors. The specific values are obtained by using already existing constants. One such constant is the angle encoder-value where the packages are sealed. From this value the angle encoder-values for “hit” and the heavy sectors are calculated.

In order to verify that the Ladder implementation was correct, simulations were made inside the PLC. By comparing a scenario simulated both in the PLC and Simulink, it could be verified that the implementation was as intended. This comparison was needed since the used PLC does not provide any kind of verification tool with sufficient resolution.

4.3 Verification of Control System

The aim with this test was to verify that the Ladder implementation of the control system is able to perform its task in reality. Ideally the algorithm should remove all collisions between more than two machines.

Unfortunately there were only two real machines available so the other (four) had to be simulated within the PLC. Since the real machines were of different production rates, a frequency inverter was installed in the faster machine so it could be given the same base speed as the slower. This was done to get the most out of the test.

Both machines were connected to an external PLC in which the control system was implemented. In the test the simulated machines were made completely deterministic, meaning that their periods were always the same and the delays were always exactly 100 ms long. This was however not the case with the real machines. As observed earlier they had two different periods, depending on where they were in the production cycle. The machine which was used during the previous tests still had a rather small difference of 3 ms, but the other machine had a difference of about 10 ms. In addition to this asymmetric behaviour there was a random disturbance. Since this was discovered during testing all that could be done was to build the mean value (from two periods). Additional uncertainties are introduced when the real machines perform delays; mostly because the break time was not exactly tuned. Both these sources of disturbances are further discussed in Chapter 5. It should be mentioned that the break times differed largely; 295 ms in the machine slowed down and 495 ms in the other. The most part of this difference can be explained by the use of frequency inverter, which instead of breaking the current as intended set the frequency to zero giving a braking force.

To evaluate the test, eight signals from the external PLC were measured. Six of them gave a pulse every time respective machine hit. The other two were used to indicate when collisions were detected. To keep the size of data on a reasonable level each measurement was restricted to 10 minutes. When no algorithm was used there were around 390 collisions (100 ms) between three machines. With use of both algorithms the different measurements showed that the number of collisions was reduced to approximately 15. The results from one of the measurements are shown by the histogram in Figure 4.6. The histogram is based on the time differences illustrated in Figure 4.7. Have in mind that all intervals under 100 ms are collisions. As seen, the simulation has no collision (except for a transient). The measurement from the real test shows a number of collisions. As stated above, this is due to uncertainty in both period and delay for the real machines. It also can be observed that the number of large time intervals is decreased when control is performed, which of course is a consequence from removing the collisions. Another obvious consequence, proved in the test, is that a decreased number of machines results in a decreased number of collisions.

Also another aspect of the program was tested. A few times during the test a new package material reel was inserted. This gave a perfect opportunity to see that the program can handle the situation when a machine is out of production.

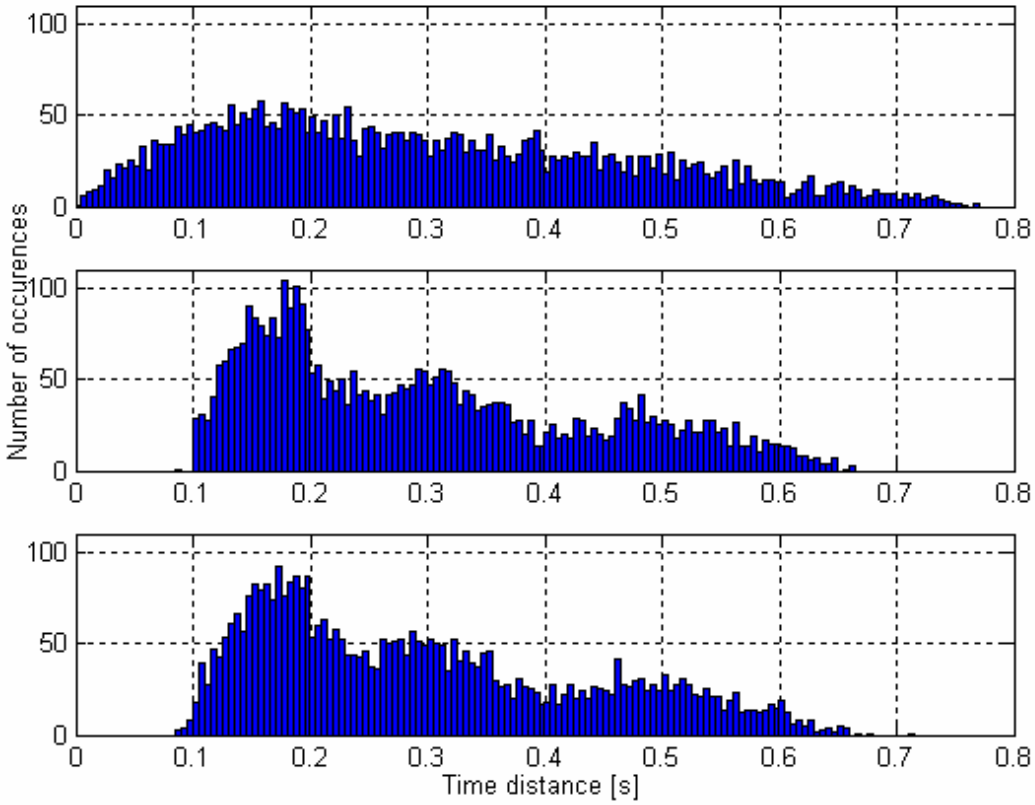


Figure 4.6 Histogram over time differences between hits from a test without control (top), a simulation in Simulink with control (middle) and a test with control on full scale machines (bottom). Every bar represents an interval of 5 ms.

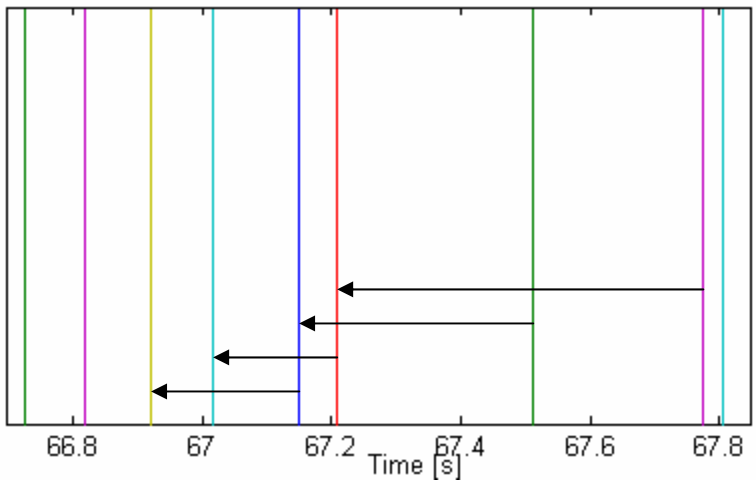


Figure 4.7 Time differences between the hits from the different machines (different colours) which the histogram is based upon.

Chapter 5

Experiences and Conclusions

This thesis has investigated different possibilities to make parallel filling machines non-concurrent. In this chapter the results are summarized and discussed. Suggestions on further development are also given.

5.1 Summary

In Chapter 2 different possibilities to affect the production rate of a machine were investigated. Immediately it was determined that it is the machine's main motor that has to be affected to adjust the production rate. Two different methods were considered realistic and suitable to use.

The first method implies that the current to the main motor is broken for short time intervals. During this time the motor has no driving force, which naturally leads to decreased rotation speed. For the fulfilment of this method, installation of a solid-state relay is required. The other method uses a frequency inverter to affect the rotation speed. In contrast to a solid-state relay, which can only make sporadic delays, a frequency inverter can really change rotation speed of the motor. Furthermore, a frequency inverter can both increase and decrease the rotation speed which is a great advantage compared to solid-state relay. As mentioned in Section 2.1.5 Tetra Pak considered the solid-state relay to be the most interesting alternative, mainly because of the lower cost.

Furthermore, it was determined that the control program should be placed in an external PLC. The reason is that the PLCs in the machines do not support network communication.

In Chapter 3 several algorithms were developed in order to perform the control, either with solid-state relay or frequency inverter. The main focus has been to develop algorithms for a solid-state relay.

All algorithms were, when developed, seen as possible solutions. However, simulations showed that some algorithms decreased the production rate too much. The decrease in production rate was so heavy that the most reasonable approach was to lower the demands of avoiding all collisions. This resulted in *Algorithm V – Allow 2-hits*, which only avoids collisions between three machines' hits. Compared to the other algorithms the decrease in production rate is considerably smaller. Since the algorithm only removes collision between three machines, it is of course possible for machines with same base speed to fall into step. This is solved by combining it with another algorithm that controls machines with same base speed. The algorithm considered to do this best, regarding production rate, is *Algorithm II – Fox jump*. Thus, it is the combination of *Algorithm V – Allow 2-hits* and *Algorithm II – Fox jump* that is the foundation of the control program. The properties of the different algorithms were presented in Section 3.3.

Chapter 4 first presents procedure and results from a test on the full scale machine. The test showed that it is possible to extend the period as long as suggested from the simulations.

Further, experiences from the implementation in Ladder Diagram and the results from a full scale test are discussed. The purpose of the test was to verify that the control system worked in reality. Unfortunately only two machines could be used for the test, the remaining four machines were simulated in the PLC internally.

5.2 Discussion

At first sight the problem of concurrent hits did not seem to cause any particular difficulties. According to the description given by Tetra Pak concurrency was something that occurred with minutes in between. The simulation model could however both confirm and deny this description of the problem. In the case where all machines have the same base speed it is correct that it takes several minutes before concurrency is seen, but if the machines are of different base speeds it is a matter of seconds. Another detail, not fully realized at the beginning of this project, was the increase in complexity for every new machine connected to the system. Without difficulties one can imagine how hits from two machines can collide with each other. To some extent one can also imagine the different scenarios between hits from three machines, but when more machines are added the scenarios are unimaginable. Here the possibilities of simulating proved to be invaluable.

Originally, the solution algorithms were designed with aim at avoiding all collisions. However, the simulations showed a decrease of approximately 5% in production rate when doing so, which of course is an unacceptable deterioration. Therefore the decision was taken to allow presence of some collisions if it gave reasonable production rate. The result from this was to allow collisions between two machines with different base speeds. With this measure the decrease in production rate was improved to be between 0-0.5% in the normal case. Some simulations have shown scenarios where a machine has had a decreased production rate of up to 1%. This is however an extreme case which from a statistical point of view ought to be levelled during a longer time. Since TFA/3 has an over-capacity, it will not fall below its lowest stated production rate.

It should be emphasized that the selected solution algorithm does not use any master. This means, no machine rules the others. It is instead the positions of the hits in every predicted collision that determine whether a machine should make a delay.

The reason why an algorithm for frequency inverter was developed, despite solid-state relay is the main alternative, was to explore its possibilities. Because of the limited time spent, the algorithm is not fully developed. It distributes hits from machines with same base speed equally, but the possibilities to avoid collisions between machines with different base speeds are not investigated. Still it was enough to realize the advantages from *both* being able to increase and decrease production rate.

When tests on the full scale machines were performed, two real machines were available. As mentioned in Section 4.3 two unforeseen properties showed.

The first one was that the machines' periods were not always the same. For one machine there was a clear asymmetric behaviour. The other had a more random disturbance. The asymmetry can be illustrated by thinking of the production cycle as a circle. Every hit is the end of one period but also the beginning of the next. This means that if one period is a little longer the next one will be slightly shorter. To minimize this problem the mean value was calculated. Further testing is required to determine what the best action is. If, for example, the asymmetry is more common the most suitable would be to keep track of two periods per machine.

The other property showed that the machines had to break the current for different lengths of time in order to obtain the same delay. For the performance of the solution it is important that the delay is close to the intended one. For the two real machines in the test the difference in break time was approximately 200 ms. As stated in Section 4.3 the main part of this difference can be explained by the use of the frequency inverter. Regardless of the reasons, it showed to have an influence on the performance and is therefore something that ought to be paid attention to. Our suggestion is that every machine keeps track of its period time and adjusts the break time until the delay is 100 ms (or at least very close to it). This should be possible with simple proportional control. We see this as a demand before the solution could be regarded as fully developed.

The main aim of this thesis was to remove all collisions in a multiple filling machine installation. However, this had a huge negative effect on production rate. After discussions with Tetra Pak the aim was therefore adjusted to also take the production rate into consideration and consequently accept some collisions. This was necessary in order to obtain a reasonable solution using a solid-state relay. Also the demand to handle different package types was removed after discussion with Tetra Pak.

5.3 Future Potentials

Besides the need of adaptive length of the current breaks mentioned above, the following issues should be considered for future development.

Since the solution has been built to fit TFA/3, it is written to handle only two different base speeds and to be put into an external PLC. If the solution is to be used on other systems with more base speeds, changes must be made. It would naturally be desirable to be able to manage an unlimited number of base speeds. When it comes to where the program should be implemented, the best solution would be if it were placed in the PLC of every machine. For TFA/3 this was impossible because of the lack of network support, but for other systems the situation may be different.

Further development of the control system would naturally be to remove all collisions without decreasing the production rate. As mentioned before this is considered to be possible only if a frequency inverter is used. However, we are not convinced that even a frequency inverter could manage, but simulations of *Algorithm VI – Equal distribution (frequency inverter)* shows promising results. Another advantage is that the production rate for the machines can be increased. The only drawback for the frequency inverter is the higher price compared to solid-state relay. From all other aspects a frequency inverter is preferable.

Chapter 6

Bibliography

Alfredsson, Alf – Jacobsson, Karl-Axel – Rejminger, Anders – Sinner, Bengt (1986), *Elkraftteknisk handbok 2 – Elmaskiner*, Esselte Studium, Uppsala

Barnes, Lee – Hardin, John – Gross, Charles A. – Wasson, Dewain (1993), An Eddy Current Braking System, *Proceedings SSST '93 Twenty-Fifth Southeastern Symposium on System Theory*, p. 58-62

Bishop, Anthony (1986), *Solid-State Relay Handbook with Applications*, Howard W. Sams & Co., Indianapolis

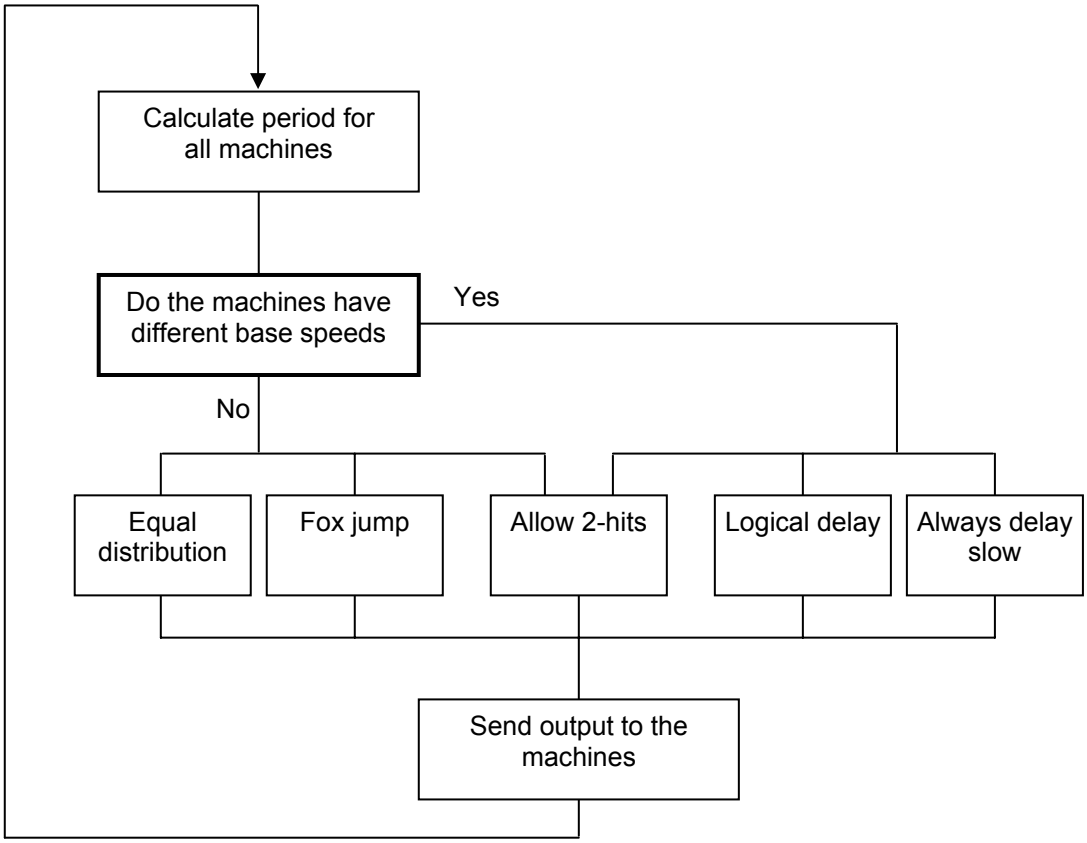
Herman, Stephen L. & Alerich, Walter N. (1993), *Industrial Motor Control – Third edition*, Delmar Publishers Inc., Albany

Lee, Kapjin & Park, Kyihwan (1999), Optimal robust control of a contactless brake system using an eddy current, *Mechatronics*, 9, No. 6, p. 615-631

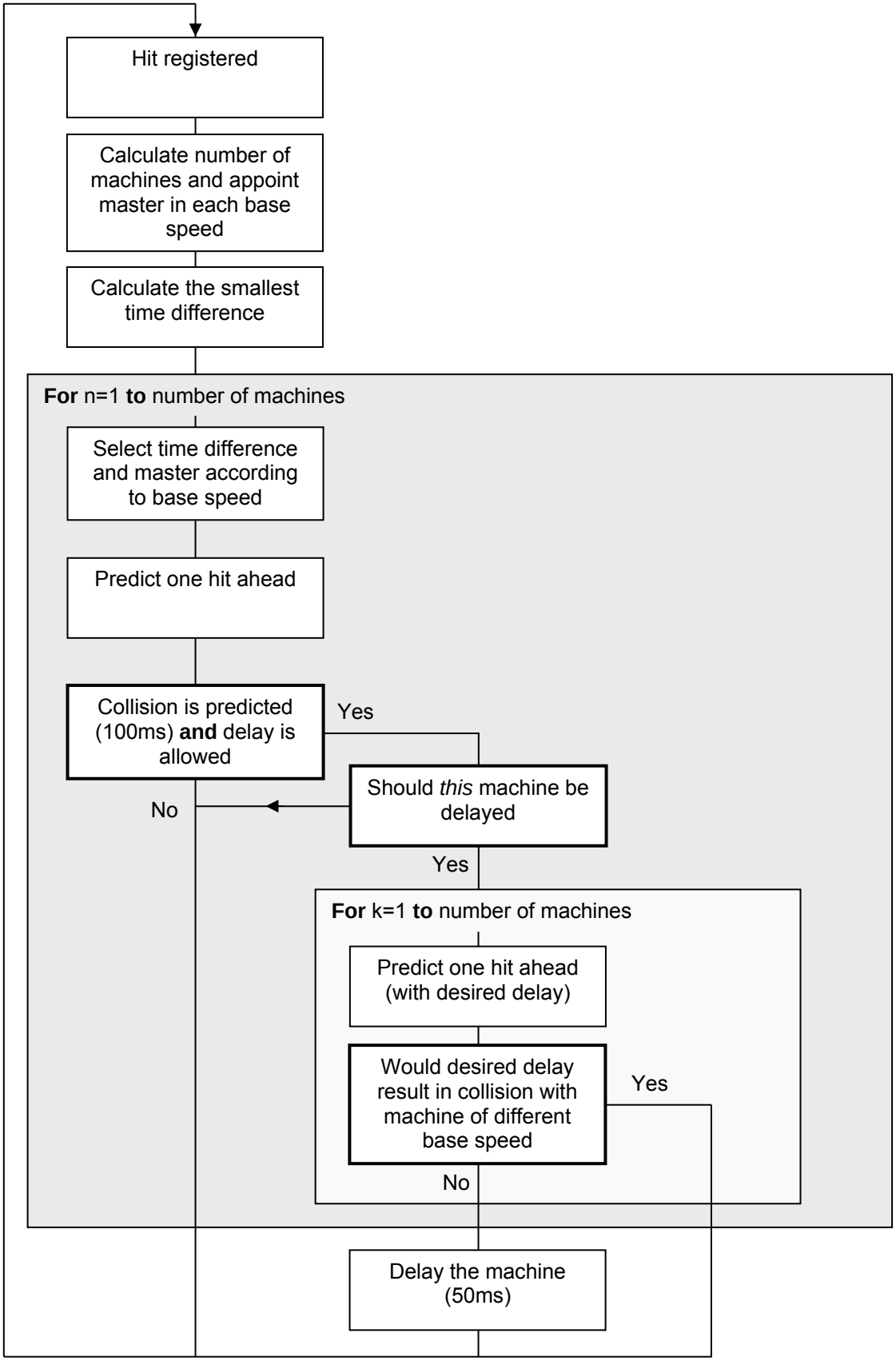
Tetra Pak Carton Ambient AB (2006-03-05), *Tetra Fino Aseptic, Pure Simplicity*, http://www.tetrapak.com/docs/2160en_Fino_1low.pdf

Block Diagram

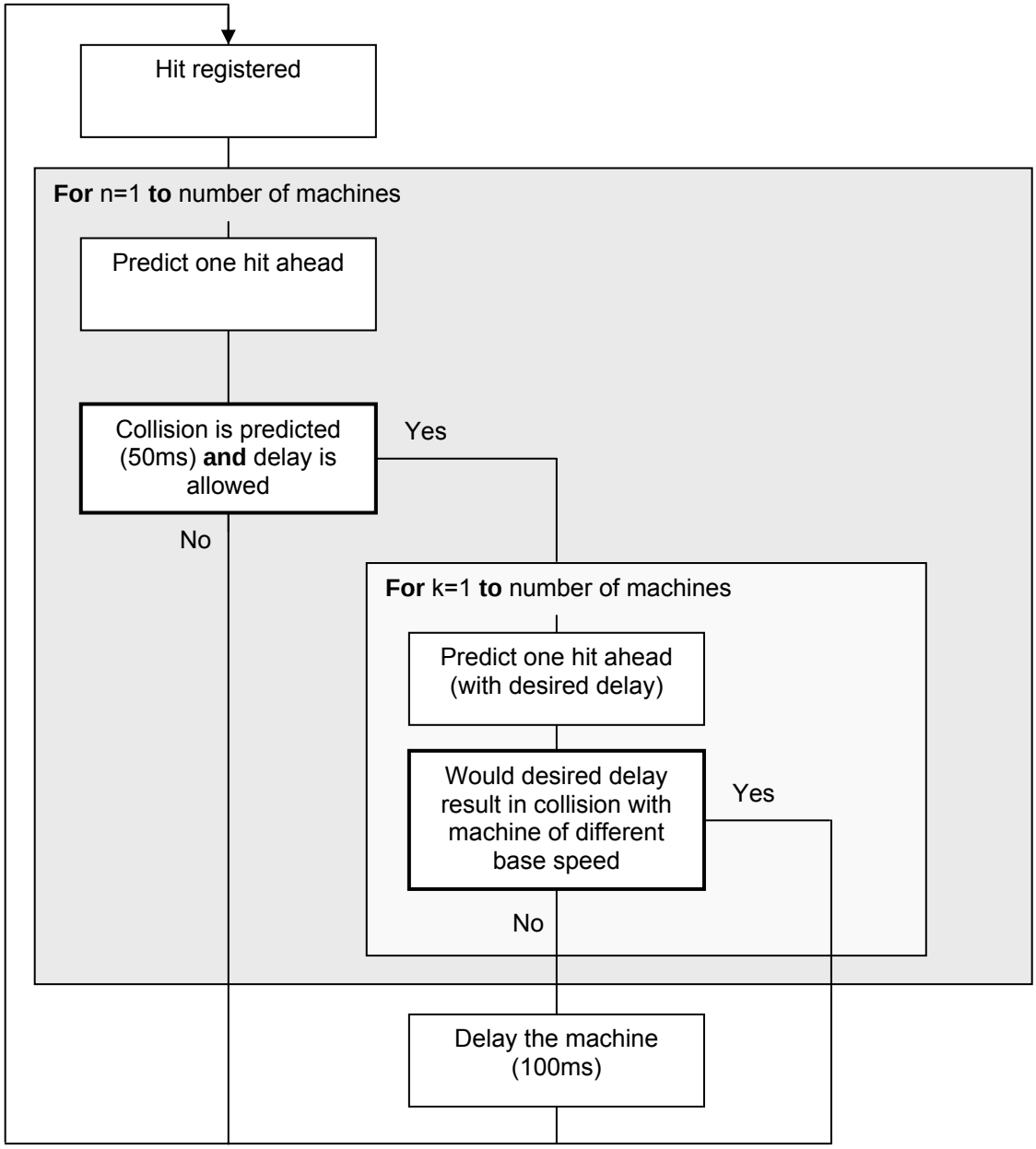
A.1 Main Program for Solid-State Relay



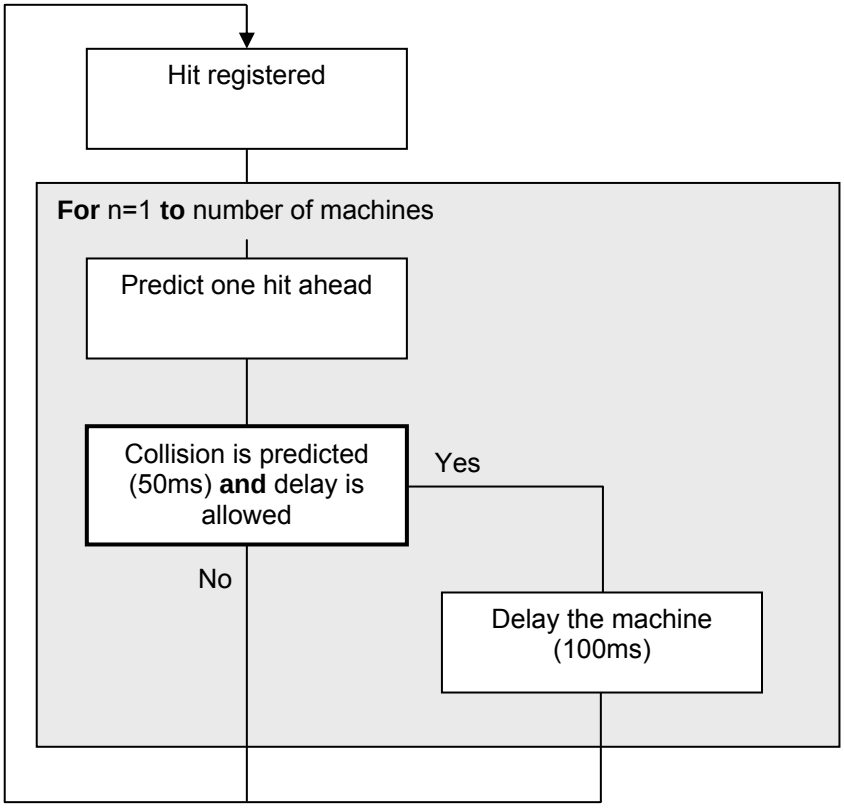
A.2 Algorithm I – Equal distribution



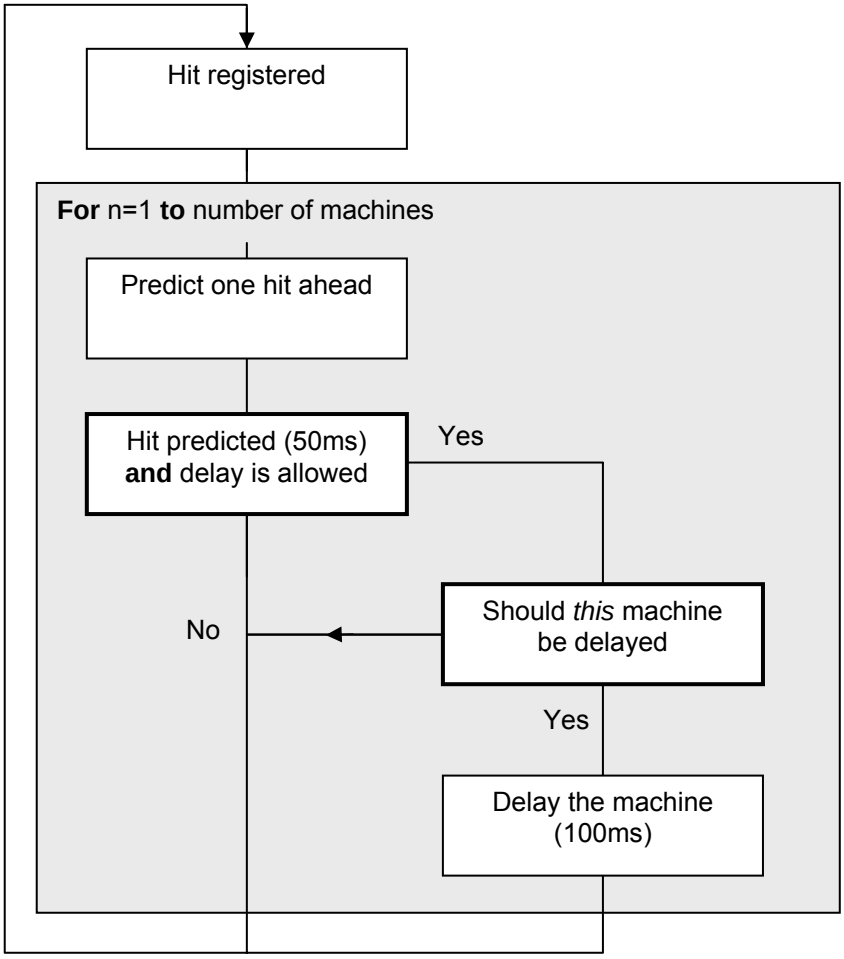
A.3 Algorithm II – Fox jump



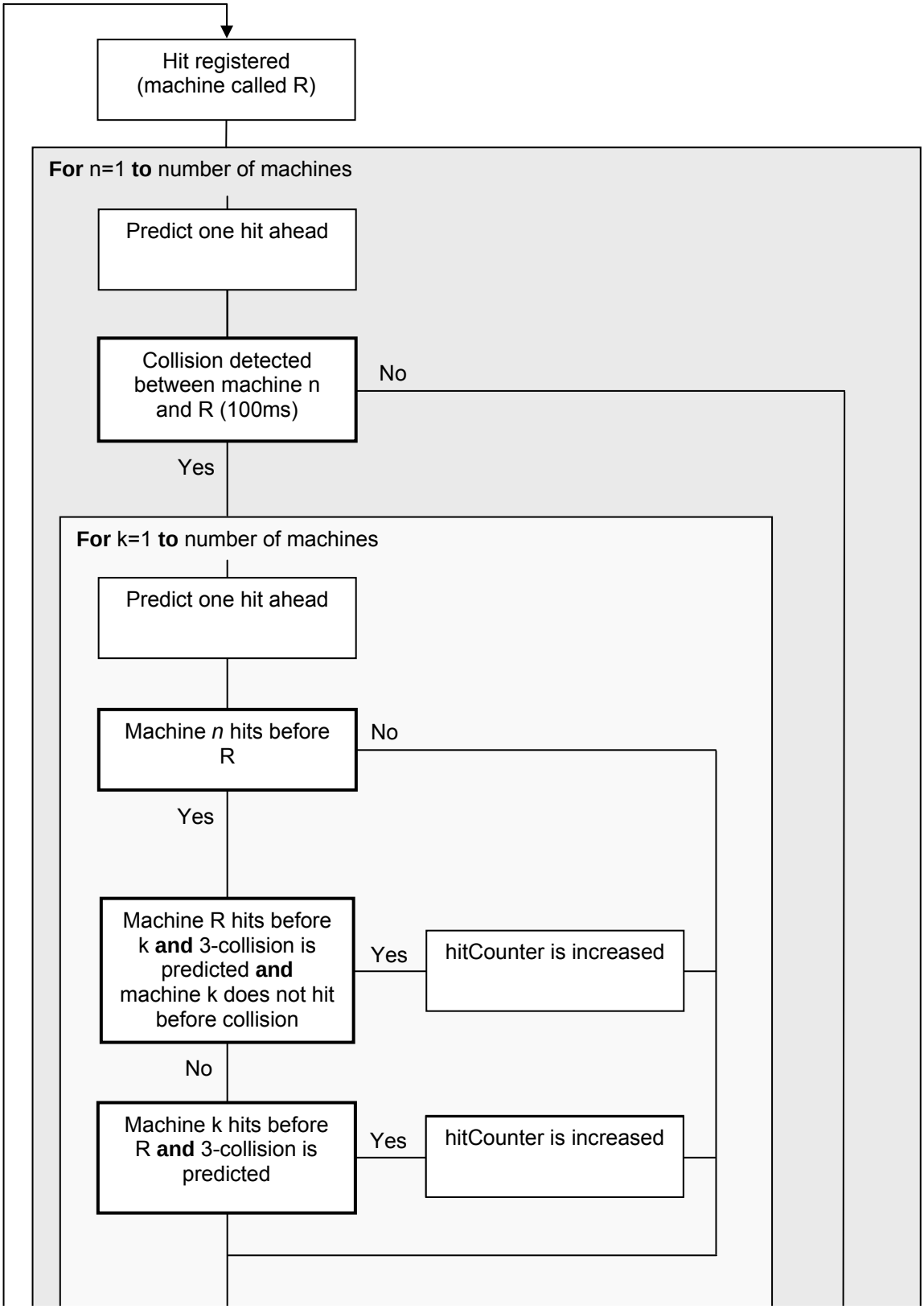
A.4 Algorithm III – Always delay slow

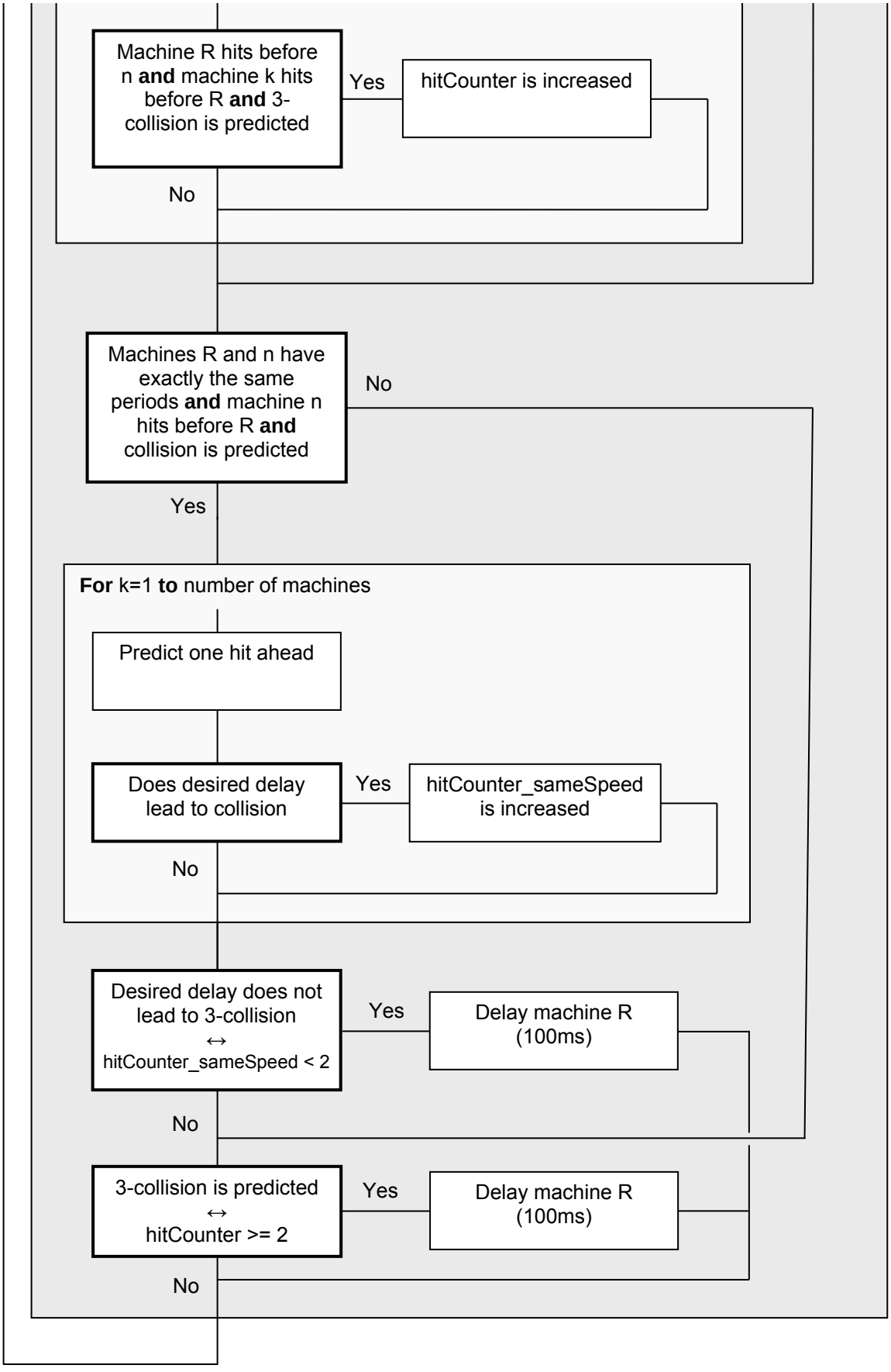


A.5 Algorithm IV – Logical delay

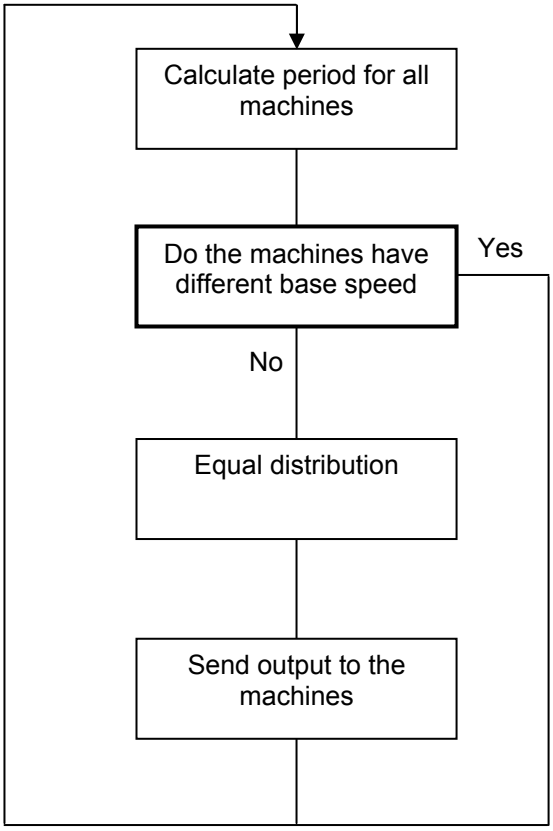


A.6 Algorithm V – Allow 2-hits

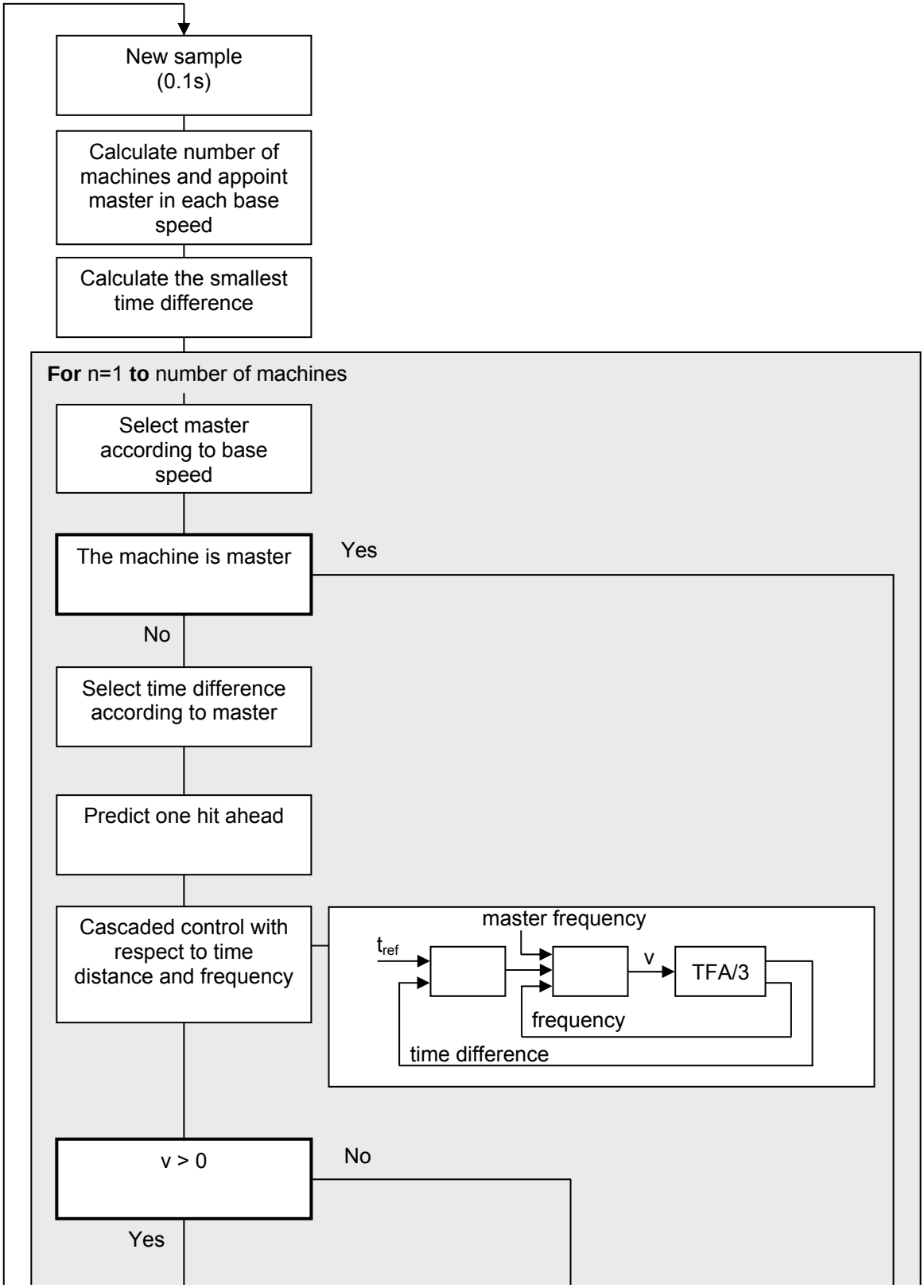


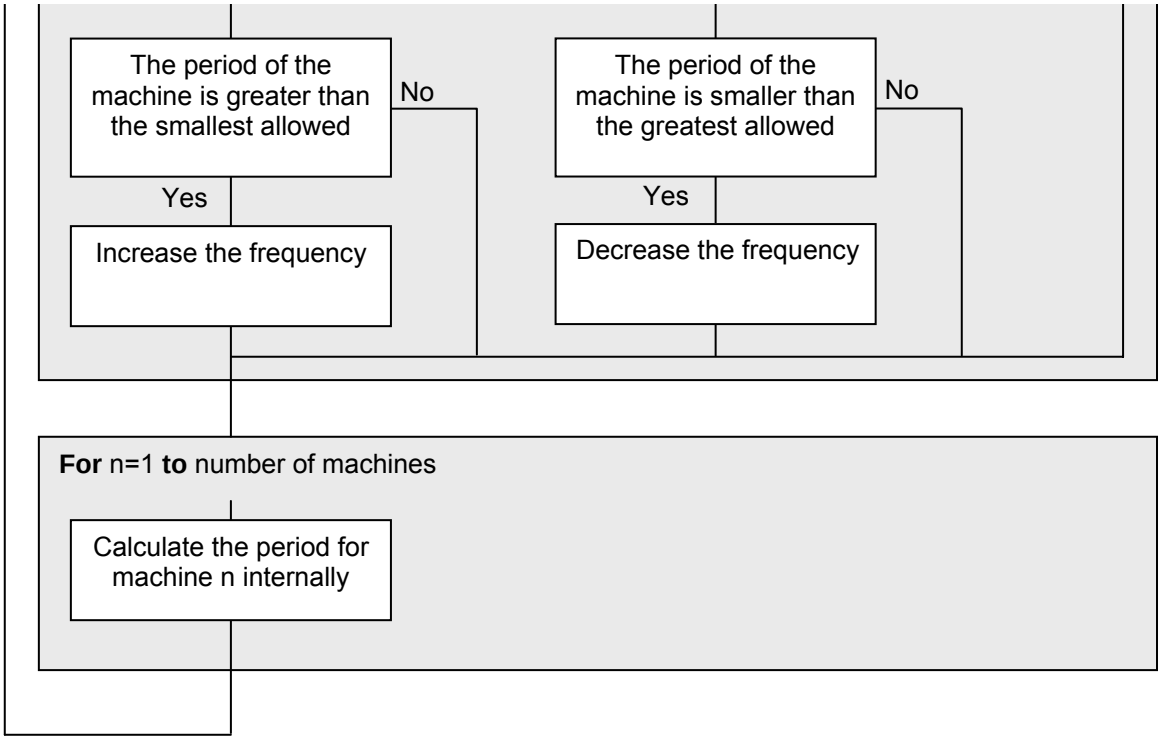

Machine R hits before n **and** machine k hits before R **and** 3-collision is predicted
Yes
hitCounter is increased
No
Machines R and n have exactly the same periods **and** machine n hits before R **and** collision is predicted
No
Yes
For k=1 **to** number of machines
Predict one hit ahead
Does desired delay lead to collision
Yes
hitCounter_sameSpeed is increased
No
Desired delay does not lead to 3-collision ↔ hitCounter_sameSpeed < 2
Yes
Delay machine R (100ms)
No
3-collision is predicted ↔ hitCounter >= 2
Yes
Delay machine R (100ms)
No

A.7 Main Program for Frequency Inverter



A.8 Algorithm VI – Equal distribution (frequency inverter)




The period of the machine is greater than the smallest allowed
No
Yes
Increase the frequency
The period of the machine is smaller than the greatest allowed
No
Yes
Decrease the frequency
For n=1 to number of machines
Calculate the period for machine n internally