POLITECNICO DI MILANO
Dipartimento di Elettronica, Informazione e Bioingegneria
Master of Science in AUTOMATION AND CONTROL ENGINEERING

OPTIMAL SCHEDULING FOR ROBOTIZED PICK AND PLACE PACKAGING SYSTEMS

Graduation Thesis of:

ATEF BOUCHRIT
Mat. 814868

Supervisor*:*
Prof. Luca Ferrarini

Co- Supervisor*:*
Eng. Emiliano Pizzi

Academic year 2015-2016

ACKNOWLEDGMENTS

My special thanks go to the Polytechnic of Milan for being part of my personal and professional growth as an individual. I would like also to thank the DAISY lab team for being of constant support throughout the lifetime of this work. I would like also to thank my supervisor Professor Luca Ferrarini and my Co-supervisor Engineer Emiliano Pizzi for their constant provision.

بعد شكري **لله**، أود أن أتقدم بالشكر لوالديّ، **المنصِف بوشريط وعائشة رشيد**، اللّذان بذلا جهدا جبارًا لدعمي في جميع مراحل الحياة التي أدت إلى هذه المرحلة التعليمية. وأود خاصة أن أشكر شقيقتي، **درصاف**، لدعمي لإكمال درجة الماجستير. ويجب ألا أنسى شكر أشقّائي، **إنصاف وواصف وأحمد**، لدعمهم المعنوي.

ABSTRACT

Production systems come in diverse forms of layouts and configurations, all with the aim of increasing productivity and plant efficiency. As the system grows in size, understanding the interaction of the systems' parameters and variables becomes far from being an intuitive task. For this reason, system providers often offer software tools to simulate to a high degree of confidence the system dynamics. These tools are easy to use when considering sizing the plant to satisfy different requirements and specifications of that particular plant. However, these tools are complex and time consuming to use when it comes to developing and testing new control algorithms.

Robotized Pick-and-Place systems are a subset of production systems and have the same previous considerations. These systems consist of conveying mechanisms and more than one robot to perform the picking and placing task. This requires a carefully designed supervisor that assigns the sequence of pieces and boxes to the team of robots.

The aim of this thesis is to develop both online and offline solutions that assure a correct manipulation of products and boxes. The online solution is formalized as Mixed Integer Linear Program (MILP) and applied to the plant as a model predictive controller (MPC) with a receding horizon. The MILP problem has been developed with two lines of thoughts. One follows the idea of solving the assignment problem, and the other solves a network flow problem. Both of these solutions result in a superior solution to the banal approach adopted in many industrial solutions. At this stage, the solutions are computationally heavy to be feasible in an industrial application. However, the results of the optimal solutions have led to developing an offline solution following a simulation based optimization technique. This technique considers a widely-adopted control strategy and enhance it by manipulating a number of plant parameters. The best solution is then applied to the plant. The superiority of this optimization technique lies in the fact that it can easily be applied to different plant sizes and configuration.

Table of Contents

1 Introduction 9
1.1 Robotized packaging systems 11
2 State of the art 13
2.1 Introduction 13
2.2 Metaheuristic approach 14
2.2.1 Ant colony optimization 14
2.2.2 Genetic algorithm 14
2.2.3 Particle swarm optimization 15
2.3 Murdoch assignment algorithm 15
2.4 Broadcast of local eligibility algorithm 15
2.5 Parker's MRTA architecture L-ALLIANCE 16
2.6 The Hungarian algorithm 16
2.7 Decentralized objective function 18
2.7.1 Non-cooperative game theory 18
2.7.2 Auction algorithm 18
2.7.3 Optimal market-based task allocation via strategic pricing 19
2.8 Mixed integer linear program (MILP) 19
2.9 Conclusion 19
3 Simplified modelling for mono grip robot plants 21
3.1 Plant description 21
3.2 Model assumptions 21
3.3 Plant model 23
3.3.1 Products and pick conveyor model 23
3.3.2 Boxes and place conveyor model 24
3.3.3 Robot model 25
3.4 Model verification 27
3.4.1 Cyclic assignment 28
3.4.2 Start-up transient and steady state conditions 29
3.4.3 Plant layouts 30
3.4.3.1 Model 1 31
3.4.3.2 Model 2 32
3.4.3.3 Model 3 33
3.5 Conclusion 34
4 MILP formulation of the assignment problem 35
4.1 Introduction 35
4.2 Future mission approach 36
4.2.1 Introduction 36
4.2.2 Problem parameters 36
4.2.3 Decision variables 36
4.2.4 Objective function 37
4.2.5 Problem constraints 37
4.2.5.1 Work area constraints 38
4.2.5.2 Time continuity constraints 38
4.2.5.3 Special order sets 39
4.2.5.4 Maximum velocity change constraint 39
4.2.5.5 Multiplication constraints 39
4.2.5.6 Mission in progress constraints 40
4.2.5.7 Products and boxes balance constraint 41

4.2.5.8 Upper and lower bounds 41
4.2.6 Simulations and results 41
4.2.7 Conclusion 45
5 Network flow approach 46
5.1 Introduction 46
5.2 Problem parameters 47
5.3 Decision variables 48
5.4 Objective function 48
5.5 Problem constraints 48
5.5.1 Special order sets 49
5.5.2 Network flow constraints 49
5.5.3 Time window constraint 49
5.5.4 Precedence constraints 49
5.5.5 Mission in progress constraints (MIP) 50
5.5.6 Upper and lower bounds 51
5.6 Simulations and results 52
5.7 Conclusion 55
6 Simulation based optimization 56
6.1 Introduction 56
6.2 Cyclic assignment solution 57
6.3 Optimization over IBD 60
6.4 Optimization over the Switching Instance (SI) 61
6.5 Combining SI and IBD solutions 62
6.5.1 Estimation of IBD 65
6.6 Simulations and results 66
6.6.1 Numerical parameters 66
6.6.2 Effect of the distance between two boxes (Δxb) 67
6.6.3 Optimizing over IBD and SI 68
6.7 Conclusion 71
7 Comparisons and results 72
7.1 Introduction 72
7.2 Comparisons in terms of lost Products in start-up transient 72
7.3 Comparisons in terms of computational cost 73
7.4 Superiority of SbO over MILP solution 74
7.5 Conclusion 75
8 Extensions to multiple-grips robot plants 76
8.1 Introduction 76
8.2 Robot model 76
8.3 Model verification 77
8.4 Control strategy 79
8.5 Conclusion 81
9 Conclusions and final remarks 82
References 83

List of Figures

Figure 1-1: Product flow in the food industry 10
Figure 1-2: A typical plant layout 12
Figure 3-1: A three-robot plant with two rows of products and boxes with two layers 22
Figure 3-2: Pickup conveyor 23
Figure 3-3: Product state diagram 24
Figure 3-4: Place conveyor 24
Figure 3-5: Box automaton 25
Figure 3-6: Mono grip robot automaton 26
Figure 3-7: Cyclic assignment of products to a team of three robots 28
Figure 3-8: Cyclic assignment of boxes to a team of three robots 28
Figure 3-9: A representation of a cyclic assignment with zero shift, a plus sign means that the product in the x-axis is handled by the robot in the y-axis 29
Figure 3-10:Start-up and steady state conditions 30
Figure 3-11 Time between two pickups for mono grip model 1 (Simplified Model) 32
Figure 3-12 Time between two pickups for mono grip model 1 (RST) 32
Figure 3-13: Time between two pickups for mono grip model 2 (Simplified Model) 33
Figure 3-14: Time between two pickups for mono grip model 2 (RST) 33
Figure 3-15: Time between two pickups for mono grip model 3 (Simplified Model) 34
Figure 3-16: Time between two pickups for mono grip model 3 (RST) 34
Figure 4-1: Packaging system in a feedback configuration 35
Figure 4-2: Linear weighting decay 37
Figure 4-3: CI-OLM 42
Figure 4-4: Computational cost of I-OLM (left), Computational cost of CI-OLM (right) 42
Figure 4-5: Vpl profile considering I-OLM (left) and CI-OLM (right) 43
Figure 4-6: Products' assignment: considering I-OLM 44
Figure 4-7: Products' assignment: considering CI-OLM solution 44
Figure 4-8: Time between two pickups 44
Figure 5-1: Three robot plant model 47
Figure 5-2: Solver cost (Solution 1) 52
Figure 5-3: Solver cost (Solution 2) 53
Figure 5-4: Place conveyor belt velocity profile (Solution 1) 53
Figure 5-5: Place conveyor belt velocity profile (Solution 2) 53
Figure 5-6: Time between two pickups (Solution 1) 54
Figure 5-7: Time between two pickups (Solution 2) 54
Figure 5-8: Products' assignment (Solution 1) 54
Figure 5-9: Products' assignment (Solution 2) 55
Figure 6-1: Finding the optimal solution of a "free" parameter of a generic plant 57
Figure 6-2: Equal number of rows and layers (Case 1) 58
Figure 6-3: Number of rows is more than the number of layers (Case 2) 59
Figure 6-4: The number of layers is more than the number of rows (Case 3) 59
Figure 6-5: Initial Box Distance (IBD) 60
Figure 6-6: Assigning the first n products in a greedy manner 62
Figure 6-7: Algorithm to optimize generic plant layouts 64
Figure 6-8: Lost products versus (IBD,SI) 65
Figure 6-9: IBD* is depicted along with the meeting point X_{mp} 65
Figure 6-10: effect of Δxb on the number of lost products 68
Figure 6-11: Optimization over IBD and SI with Δxb= 45 [cm] 69
Figure 6-12: The arrows correspond to the solution path found by the algorithm 69

Figure 6-13: Optimization over IBD and SI with Δxb= 55 [cm] 70
Figure 6-14: Optimization over IBD (Model 1) 70
Figure 6-15: Optimization over IBD (Model 2) 71
Figure 7-1: Number of holes is more than the number of products in the system, (Model 1.1) 74
Figure 7-2: Number of holes is more than the number of products in the system, (Model 1.2) 74
Figure 7-3: Time between two pickups (SbO, Model 1) 75
Figure 7-4: Time between two pickups (MILP, Model 1) 75
Figure 8-1: Multiple-grips robot automaton 76
Figure 8-2: Validtaion in terms of the behaviour of the number of lost prodcuts 78
Figure 8-3 Time between transferring two groups of products for multi grip model 4 (RST) 78
Figure 8-4: Time between transferring two groups of products for multi grip model 4 (Extended Model) 79
Figure 8-5: Optimization over IBD 80
Figure 8-6: Optimization over the number of greedy products 80
Figure 8-7: Optimizing over SI with delta Δxb equals to 55[cm] 81

List of Tables

Table 1: Model paramters 22
Table 2: Cyclic schedule with different phases 29
Table 3: Plant layouts 31
Table 4: Prodcut distribution among robots 43
Table 5: Comparing scheduler performance in terms of the number of lost products 45
Table 6: Product distribution 54
Table 7: Comparing scheduler performance in terms of the number of lost products 55
Table 8: numerical values for the parameters of MODEL 3 66
Table 9: IBD* with Δxb of 45 [cm] and IBD^{new} computed with Δxb of 55 [cm] 70
Table 10: Comparison in terms of the total number of lost products 73
Table 11: Computational cost for finding the optimal solution on Model 1 73
Table 12: Model parameter for both simplified and extended models 77

1 Introduction

Automation is concerned with the process of reducing human interaction with industrial systems. One benefit of reducing human interaction is related to the safety and health of operators while the second one is related to increasing efficiency and reducing human errors. Industrial systems consist of several sub-systems, each with its own complexity and considerations. In one plant, one could find thermal, mechanical, electrical and hydraulic systems. All of which are working together in order to accomplish a final objective. Some of these objectives are accomplishing high speed and high precision. These objectives are hard to accomplish with no automatic procedures. In these procedures, one could focus on the design of precise tools and mechanisms. Other could work on the electronic board development that runs the mechanism. One also could contribute by developing simulation tools to study the process with minimum cost. A broader problem concerns with managing the interaction of various sub-systems and finding the optimal layout to increase efficiency and productivity. Others could focus on the scheduling and task allocation problems. Depending on the industry, each of these problems could be studied with different emphasis. Nonetheless, all of these problems are interrelated at one level or another.

Dealing with all the pre-mentioned problems regarding all industrial plants are far from being sane. Therefore, narrowing the problem into a specific type of industry is important. Over the past century, the global food market has experienced a colossal growth on demand which led to the need of increasing supplies to satisfy such a demand. Increasing supplies implies the need for producing the maximum possible number of products in the shortest time. Moreover, the need for managing and delivering supplies to the whole globe is far from being banal. The process for this type of industry starts from the basic definition of ingredients of a product finishing to the end-user. The following figure depicts the products' workflow in a food industry.

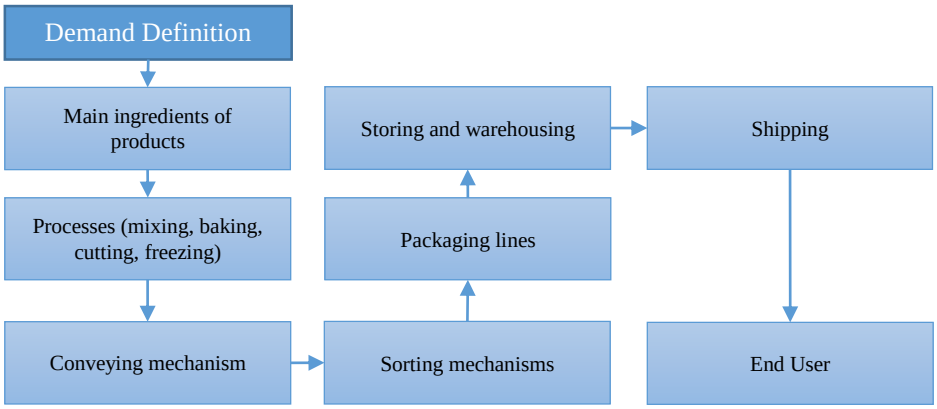

Figure 1-1: Product flow in the food industry

Starting from the market research where different aspects of the demand is defined. Depending on the size of this demand, the plant and its sub-systems are built accordingly.

The first stage involves the definition of products in terms of what goes into producing them. As an analogy to the general production systems, this stage corresponds to the definition of raw materials. Different processes is later defined. The automated system to perform these processes should respect certain requirements and constraints. At this stage, the automated system should satisfy rigorous constraints in terms of accurate and precise mixing of materials. It could also be in terms of precise temperature or pressure regulation systems. It could also be in terms of proper isolation from the surroundings. Tackling all of these problems entails an in-depth study of each aspect of the processes. A later stage worries about the apt way of transferring final products to the packaging lines. Moreover, plants are made to handle multiple types and shapes of products which requires a sorting mechanism.

Next comes the problem of packaging. There exists a hand-full of solutions and layouts to perform the packaging tasks. This is dependent on the production rate and the desired final packages. One plant could assume a high production rate with little emphasis on the orientation or the precise number of products in a package. For this type of plants there is no need for high tech manipulators such as robots. However, products more than often are required to be sorted and oriented. To respect these extra constraints, robotized packaging systems are utilized, more on this is described in section 1.1.

Warehousing, storage management and shipping are the final stages before products reach to the end user. These final stages are as important as the previous ones, and in fact without efficient warehousing and shipping the cost piles. This will define the life time of the whole system, even if the previous stages are flawless.

The aim of this introduction is to highlight the general problems involved with industrial systems. This thesis deals with the problem of robotized packaging in the food industry. Hence, a general overview of such an industry is presented. Next a thorough description of the problem is stated.

1.1 Robotized packaging systems

Robotic systems exist in different fields due to their flexibility and durability. They come in different forms depending on the intended tasks. Packaging systems are no exception due to the necessity of packing products with a certain orientation at a high rate. The packaging mechanism should also perform the task with high precision and accuracy.

Each robot has a maximum handling time of a given task. So if a plant is producing at a higher rate than a robot can handle, another robot is added. Now dealing with multiple robots requires a close attention to the allocation of tasks to each robot. For instance, it is not desirable to allocate more tasks to one robot over the others. This way a disparity of the life cycle of robots would occur. Another consideration is regarding weather or not all tasks has been fulfilled.

The packaging systems usually consist of conveying belts and robotic manipulators. Pieces, to be packed, enter from a pick conveyor belt and robots pack them into boxes on another belt. These belts can be laid out in several configurations. The flow of pieces could be perpendicular to that of boxes. The flow could also be parallel on the same or opposite direction. Each of these configurations poses its own problems and difficulties.

For this work, the parallel flow on the same direction (co-flux) is considered. Pieces are detected with a visual system as they enter the packaging system. Later, robots pick the pieces and place them into boxes. Clearly, pieces and boxes should be in the work area of robots for the packing to happen. Therefore, a scheduling strategy needs to be implemented for assigning pieces and boxes to robots.

The scheduling problem has been studied as online and offline solutions. The online solution has been formulated as a mixed integer linear program (MILP), [1], optimization and applied to the plant as Model Predictive Control with receding horizon [2]. The offline solution, however, is developed based on a mix of heuristic solutions.

Packaging lines are composed of a team of robots picking up products from a conveyor composed of multiple feeding rows and loading boxes on anther conveyor belt. Belts should intersect the work areas of the robots.

The vision system elaborates images coming from a camera. The camera is placed at the beginning of the product conveyor. It aims to detect the presence of a product and compute its precise position (further details on packaging lines in [3]). In Figure 1-2, a plant with 3 robots with a single grip (the vacuum cup can take one piece at a time) and parallel conveyors with the same direction (co-flux systems) is depicted. This plant will be the basic model to compare different scheduling strategies.

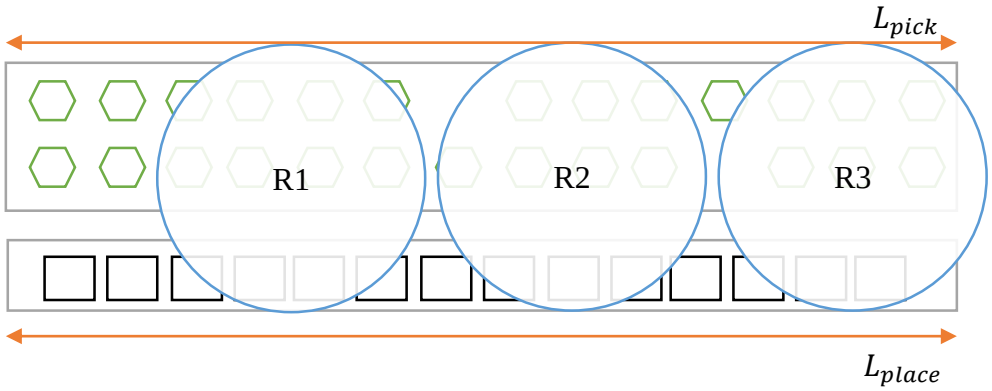

Figure 1-2: A typical plant layout

This thesis is structured into six main chapters. Chapter 2 highlights the state of the art of existing scheduling solutions. Chapter 3 presents a simplified modelling of mono grip robots. Chapter 4 and 5 discuss MILP formulations. Chapter 6 presents a heuristic solution, namely, Simulation based optimization (SbO). Chapter 7 contains a comparison between chapter 4, 5 and 7 solutions. Chapter 8 shows an extension to multi-grip robots. Finally, conclusions and final remarks are stated.

2 State of the art

2.1 Introduction

Multi-robot systems have been designed and developed for increasing a global performance of given tasks. In the narrow purpose of this work of pick-and-place multi-robots system, increasing the number of robots aims to maximize the throughput of products that successfully make it from production lines to packaging lines. The problem, henceforth, is to find the right robot-task allocation that achieves this purpose. Two main factors prevent maximizing the throughput. One is regarding the kinematic flexibility of the working robots and the second is the dynamic pick-and-place time, [4]. This work focuses on the second factor, which is of a paramount importance since it depends heavily on how to allocate tasks to each robot. In the literature, similar problems are studied under a well-known problem of Multi-Robot Task Allocation (MRTA) [5]. The study of the literature is divided into two main parts. First, heuristic solutions for pick-and-place packaging systems and second optimal assignment for multi robot task systems. The aim of this division is to look for the available solutions to the problem of pick-and-place. Moreover, understanding the general assignment problem and look for solutions currently implemented on different areas to obtain a broader insight onto the main practical limitations of such solutions.

Under the aim of sorting items on a moving conveyor belt, [4] studied the problem of efficient, defined as the throughput over the maximum throughput, picking-and-placing from flowing-items belt into specified placing areas. In [4], the authors presented robot-kinematic considerations in order to avoid limitations on efficiency due to kinematic inflexibility. For the pick-and-place time, two improved rules are presented, the FIFO and the SPT rules.

FIFO, which stands for first in first out, suggests that each robot picks the first item that it enters its working area. The improved rule considers the products that first enters the point where the robot can pick them in minimum time. [4].

The short processing time (SPT) rule suggests that each robot picks the item that can be picked in minimum time. The improved rule divides the working area of each robot to three areas, upstream, optimal and downstream areas. Therefore, the list of items to be picked in SPT are divided into items in downstream and items in upstream. Now the rule is as follows:

If the SPT objects in the upstream enter the optimal region, pick them; otherwise, pick the SPT objects in downstream area.

2.2 Metaheuristic approach

Another interesting task allocation solution is presented in [6]. The proposed idea tries to seek the right mix of multiple queuing strategies to maximize the picked products off a conveyor belt. The search algorithm implements three metaheuristic approaches, the ant colony optimization, Genetic algorithms and particle swarm optimization. Here, the three algorithms are explained.

2.2.1 Ant colony optimization

The algorithm is inspired from the behaviour of ants when searching a path between their colony and a food source. Each ant in the team of ants starts seeking a path heading to an arbitrary direction and meanwhile it leaves a trace by producing a compound known as pheromone. Once a root to a food source is constructed from different combinations of roots, the team follows this root. This idea is adopted for the multiple task-robot allocation problem, [7]. The idea of the algorithm is that an ant k selects to go through multiple queuing strategies for a robot. For example, ant k selects the FIFO rule for robot n to pick and place a number of products. The ant then calculates a fitness function, which is related to the lost products. The same ant tries another rule and computes its corresponding fitness function. Another ant does the same with respect to another robot. When all ants have computed their fitness functions, the strategy mix that minimizes the lost products will be selected.

2.2.2 Genetic algorithm

The Genetic Algorithm (GA) is inspired form the natural selection process where inheritance, mutation, selection, and crossover take place in order to create a better population. The GA as an optimization algorithm seeks randomly a set of solutions to a particular problem. The algorithm, then, eliminates solutions that are less performant and only the best set of solutions is selected. The process is iterated until a fitness function, which is an objective function for the optimization problem, is minimized or a maximum number of iterations is reached. Each iteration contains a population of solutions that is inherently better than the previous population since only the best set of solutions is selected.

The algorithm is summarized as the following:

1- Randomly select a population: in this case, a population is the set of queuing solutions
2- Evaluate the fitness function of each queuing solution
3- Keep a set of the best solutions
4- Based on the best solutions create a new population

5- Repeat this process until the fitness function is minimized or all the maximum number of generations have been reached

2.2.3 Particle swarm optimization

This algorithm as the previous ones were inspired from nature. To be specific consider a group of birds flying around looking for food. Assume also that the food is limited, which means that the birds try to reach the food faster than the others do. Now the optimal solution for a bird in the group is to follow the closest bird to the food. This means that the bird in question needs to adjust its dynamic variables (velocity and position) in order to reach the food as the closest birds. This idea is adapted to the problem of pick and place, where birds or particles are thought of as the queuing strategies for each robot. Each strategy has a position, velocity, a local best position and a global best position among all strategies. Initially, the algorithm randomly assigns a particle to robots and randomly initializes the particle's positon, velocity, local best position and global best position. The position of the solution can be interpreted as the current number of lost products. The velocity update of the solution is a function of the current position and both the local and global best positions. The algorithm iterates among all robots and these parameters are updated. The algorithm stops when a desired global best is obtained and the tasks are assigned. For further details on this implementation, see [6].

2.3 Murdoch assignment algorithm

This algorithm is implemented where the tasks are not known in advance. In other words, new events occur online and the scheduler should be able to allocate them. Under the assumption that tasks cannot be reassigned, the Murdoch algorithm out performs any other online assignment algorithm [8]. The algorithm states: "when a new task is introduced, assign it to the most fit robot that is available" [5].

2.4 Broadcast of local eligibility algorithm

This algorithm solves the online assignment problem with the following idea:

1- Search for robots if any not assigned
2- Find the highest utility for the robot task pair, otherwise quit
3- Assign the task to the robot and remove them from consideration
4- Repeat

In this algorithm, each available robot in a team defines its ability to take on a task and broadcasts its result to the rest of the team. Regarding a particular task, the task is given to the

robot that is most eligible to take on the task. If a task is taken by a robot, the task is removed from the set of tasks to be carried out by the team of robots. The robot's eligibility is measured based on a utility function defined for each robot, more details in [5].

2.5 Parker's MRTA architecture L-ALLIANCE

L-ALLIANCE is another interesting solution to the online assignment problem. L-ALLIANCE is able to learn the assignment from experience and if well-trained, the algorithm can perform better that its greedy counterpart [5].

2.6 The Hungarian algorithm

In the study of the optimal assignment problem, several primal and dual methods have been proposed. These type of algorithms assume to have a centralized objective function that needs to be either maximized or minimized. The solution is then carried out in a primal and dual manner of a linear program relaxation technique. The Hungarian algorithm is favoured due to its low computations cost, which is of $O(n^3)$. Formally, the problem is posed as the following:

The problem assumes to have a set of workers $P = \{1, \dots, n\}$ and a set of jobs $J = \{1, \dots, n\}$. The assignment problem tries to assign exactly one job to a worker and each worker have exactly one job.

$$minimize\ z = \sum_i \sum_j c_{ij} X_{ij}$$

$$s.t. \sum_i X_{ij} = 1\ \forall j$$

$$\sum_j X_{ij} = 1\ \forall i$$

$$X_{ij} \geq 1$$

Where $i \in P\ and\ j \in J$

X_{ij}: is a binary number, when the number equals to 1, the worker i is assigned to job j

c_{ij}: is the cost of assigning a worker i to a job j

This formalism is a linear program relaxation of the assignment problem, which can be solved with the simplex algorithm. In the simplex algorithm, the solution starts with introducing slack and surplus variables to change the inequality constraints to equality ones.

Then all constraints along with the objective function are organized in a tableau. The first basic feasible solutions are calculated and it stops if the optimality conditions are satisfied. Else, the algorithm starts looking for the next pivot and computes the new set of basic feasible solutions and so no. the Hungarian algorithm on the other hand works out the solution by employing the primal and dual problem of a linear program. The algorithm starts with a qualification matrix that arranges the workers and the jobs with the corresponding cost c_{ij}.

Workers/ Jobs	1	…	n
1	c_{11}	c_{1j}	c_{1n}
…	c_{i1}	c_{ij}	c_{in}
N	c_{n1}	c_{nj}	c_{nn}

The next iteration involves subtracting the costs row wise with row minimum and column wise with column minimum until each column and row have at least one zero. This step is interpreted as transforming the original problem to the dual one by introducing two variables that sums to have the same original cost, an illustration to this is given later. The assignment is made on the rows that have only one zero and cancelling the remaining zeros on the corresponding column. The resulting n assignments are the one that minimizes the overall cost. If only $n - k : k = 1,2, \dots, n$ assignments are made, the algorithm assigns new sets of variables and once again, the problem is transformed to a new dual problem. The algorithm stops when all n assignments are made. There are some situations where this algorithm fails to find the optimal solution. For these situations, there exists further extensions to the algorithm that are not presented here. An example of $n = 3$ follows:

Consider the following qualification matrix of 3x3.

$$\begin{pmatrix} 10 & 3 & 15 \\ 4 & 5 & 9 \\ 7 & 8 & 2 \end{pmatrix}$$

By subtracting the row minimum, we obtain:

$$\begin{pmatrix} & \mathrm{v}_1 = 0 & \mathrm{v}_2 = 0 & \mathrm{v}_3 = 0 \\ u_1 = 3 & 7 & 0 & 12 \\ u_2 = 4 & 0 & 1 & 5 \\ u_3 = 2 & 5 & 6 & 0 \end{pmatrix}$$

The new cost $\overline{c_{ij}} = c_{ij} - (u_i + v_j)$. If the new cost is taken to be zero, we obtain the original minimum cost $c_{ij} = (u_i + v_j)$. Further aspects of the algorithm are in [9]. When

dealing with large number of robots this centralized solution is more computational costly than other market-based optimal solutions which are decentralized.

2.7 Decentralized objective function

2.7.1 Non-cooperative game theory

The authors of [10] studied the problem of multi-robot coordination in pick-and-place tasks on a moving conveyor where robots are lined-up in series facing a moving belt of products. The idea of the solution is explained. First, the robots have the ability to communicate among each other. Moreover, each robot is an independent agent that tries to minimize its local cost function with respect to the assigned task. The implemented objective function simply states: "pick products that are available in the agent working area and that are the furthest from the agent's neighbours". This means that each agent requires information from its neighbours to construct its decision. Further details are in [10].

2.7.2 Auction algorithm

This algorithm is the most favoured algorithm in the literature of MRTA because of its decentralized nature, which decrease the computational cost [11]. The idea of the algorithm is that a main agent acts like an auctioneer and the team of robots as bidders. Each task j has a pre-determined price p_j and every robot i has a particular utility price for a task j. A profit margin (m_{ij}) is a merit calculated by each robot i to be maximized as follows:

$$m_{ij} = u_{ij} - p_j$$

If a robot prefers a certain task j, the robot bids on a task. If no other robots are bidding on that particular task, the task is assigned to the bidder. In case of conflicts, where multiple robots are bidding on a task, each bidder suggests an increment equals to the difference between its preferred profit margin and the second preferred profit margin as follows:

$$\gamma_i = v_i - w_i + \epsilon$$

Where:

$$\gamma_i \; is \; the \; increment; v_i = \max_{\forall j}\{m_{ij}\}; \mathrm{w_i} = \max_{\forall j \neq j^*}\{m_{ij}\}$$

Note that: j^* is the preferred task

ϵ is a positive number to guarantee the termination of the algorithm by preferring a bidder when two or more bidders have the same increment. This operation is referred to as

ϵ −complementary slackness. Finally, the bidder with the highest increment takes the task. This process is repeated until no assignment is left. Clearly, the medium of communication and the computations power of each robot play an important role in the success of such an algorithm. Further details on the algorithm is found in [12].

2.7.3 Optimal market-based task allocation via strategic pricing

Just like the auction algorithm, there are tasks and incentives to do these tasks. The main difference between the two algorithms is that the market is looked at from the merchant's point of view where his main purpose is to sell all items. In this algorithm the roles are changed. Now, robots are thought of as customers and the selected agent as a merchant. Once again, each customer is trying to select items that matches his needs in terms of maximizing his profit margin. In the case of conflict, the merchant increases the price of the task so that only one customer is interested. This process is repeated until all tasks are assigned. The detailed implementation of this algorithm can be found in [11] where both the centralized and the decentralized solution are treated.

2.8 Mixed integer linear program (MILP)

MILP is one of the most popular techniques when comes to modelling difficult problems as an optimal problem. In this thesis, two different versions of MILP has been derived and investigated. The motivation for this is to find the optimal plant manager and to study the feasibility of implementing them as online controllers. In case of infeasibility, the optimal solution is worth studying in terms of the definition of a proper objective function and proper problem constraints. The trade-off between computational cost and optimality is also highlighted. Further details on integer programing can be found in [1].

2.9 Conclusion

To conclude, there are several algorithms and techniques yet to be discovered to solve the problem of robot-task allocation. The solutions to the problem are either heuristic solutions or as a solution to an optimal assignment problem. Heuristic solutions perform well when developed on an ad-hoc manner and lacks generality. On the other hand, the optimal assignment problem can be generalized to large sets of robots and tasks assuming that the computations and communications power of robots are reliable and fast enough. Furthermore, the optimal assignment is studied as to maximize or minimize a centralized or decentralized objective function. The aim of decentralization is to divide the computations power all over the

team. Finally, the need for online optimization rises from the fact that multi-robot systems are dynamic and uncertain to tasks, which require online adjustments of offline solutions.

3 Simplified modelling for mono grip robot plants

3.1 Plant description

This work focuses on the study of a packaging system that consists of two parallel conveyor belts and a team of delta robots mounted in a series configuration. The team of robots receives a command from a centralized controller to pick products from one belt and pack them into boxes on the other belt. Both products and boxes flow on their designated belts in a co-direction configuration. Hereinafter, products and boxes' belts are referred to as belts A and B, respectively.

This class of plant layouts assumes a number of different arrangements. A robot in this class can have a mono or a multi-grip end-effector. Depending on the maximum throughput of pieces, one solution is preferred to the other. In this chapter, a mono-grip plant model is developed and, at a later chapter, an extension to the multi-grip case is discussed. In section 3.2, model assumptions are presented. Section 3.3 describes the development of the plant model. Finally in Section 3.4, the simplified model is verified against a more accurate plant model.

3.2 Model assumptions

The plant consists of two extended conveyor belts where products and boxes can flow. A number of robots are mounted in a series arrangement along the longitudinal axis of belts. Each robot has work-area boundaries defined with respect to the pick and place conveyors. The minimum and maximum bounds of the work area, with respect to belt A, are referred to as $L^r_{pi,min}$ and $L^r_{pi,max}$. Whereas, the bounds with respect to belt B are referred to as $L^r_{pl,min}$ and $L^r_{pl,max}$. Depending on the orientation and the position of a robot with respect to the belts, the work-area boundaries change.

Products enter the system from the pick conveyor belt in one or multiple rows (N^p_{row}). One row contains products with the same position along the cross direction of the products' conveyor (vertical axis). Just before the first position of the first robot, a vision system is placed in order to identify the products' initial position along the horizontal axis. Each product is assigned a non-decreasing identification number p, such that x_p is greater than or equal to x_{p+1}. The position of the product x_p is the relative distance of a product p, along the longitudinal axis (horizontal), measured with respect to the position of the vision system. A relevant plant parameter is the production rate (PR), which consists of the number of products that enters the packaging line in a fixed amount of time.

Boxes enter the system from a spacing mechanism onto the place conveyor belt, and hence the distance between boxes is a constant value. Like products, the boxes are assigned a non-decreasing identification number b, such that, x_b is greater than x_{b+1}, where x_b is the distance of a box b measured with respect to the entry point of the first box. The entry point of boxes is aligned with the position of the vision system. Each box can have one or multiple layers referred to as N_{layer}^{b}. The vacant places in a layer are referred to as holes. Hereinafter, N_{layer}^{b} is used to refer to a hole per box as well, since each layer contains only one product.

Each robot is considered to be a black box that performs the task of pick-and-place within a mission time. This time is the total travelling time from belt A to B and to A again. The picking and placing action is assumed to be instantaneous due to the fact that the traveling time along the z-direction is much shorter than the traveling time between belts. Finally, the model assumes that robots do not fail to pick a product nor fail to place it into a box. Figure 3-1 shows a plant consisting of three robots, two rows of products and boxes with two layers/holes. Table 1 contains a list of the model parameters with the adopted unit of measure.

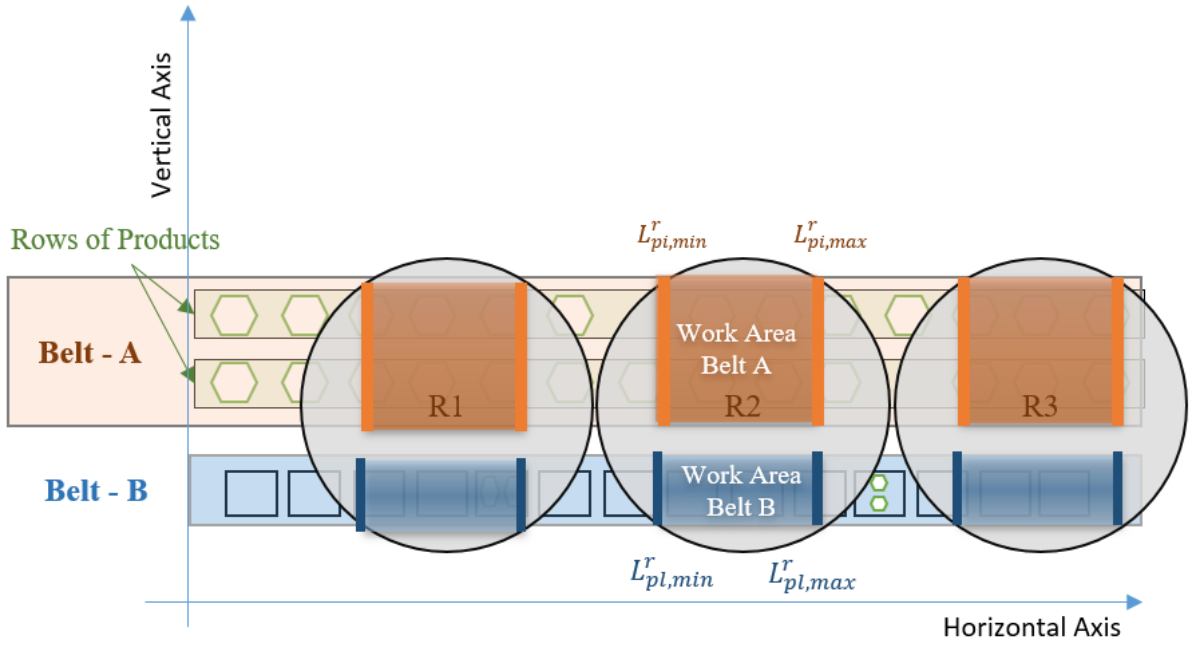

Figure 3-1: A three-robot plant with two rows of products and boxes with two layers

Table 1: Model paramters

Parameter	**Description**	**Unit of measure**
PR	Total Production Rate	Piece/min [ppm]
x_p	Product position on pickup conveyor	[m]
x_b	Box position on place conveyor	[m]
Δx_b	Distance between boxes	[m]

Δx_p	The horizontal Distance between products	[m]
Δy_p	The vertical distance between products	[m]
N_{row}^{p}	Number of Product rows	[#]
N_{layer}^{b}	Number of layer per box	[#]
T_m	Mission Time	[s]
L_{place}	Length of Place conveyor	[m]
L_{pick}	Length of Pick Conveyor	[m]
$[L_{pi,min}^{r}, L_{pi,max}^{r}]$	Max and min work area boundaries (w.r.t. Belt A)	[m]
$[L_{pl,min}^{r}, L_{pl,max}^{r}]$	Max and min work area boundaries (w.r.t. Belt B)	[m]
R	Total number of robots	[#]
P	Total number of products	[#]
B	Total number of boxes	[#]

3.3 Plant model

3.3.1 Products and pick conveyor model

The products enters the system onto the pick conveyor belt and travels at a velocity equals to V_{pi}. Given a production rate of PR, the distance between two products follow this rule:

$$\Delta x_p = N_{row}^{p} \frac{V_{pi}}{PR} \quad \textbf{(3.1)}$$

Figure 3-2: Pickup conveyor

According to the model assumptions, the distance between products along the vertical direction Δy_p is constant. Moreover, it is assumed that products along the vertical direction are always reachable by robots if along the horizontal direction a product is within the work area. A product in the system is modelled with an automaton of four states as plotted in Figure 3-3. The states and transitions of the product automaton are described below.

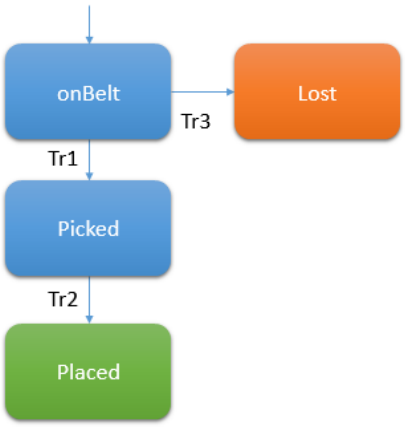

Figure 3-3: Product state diagram

States

- onBelt: the products is flowing on the pickup conveyor belt
- Picked: the product is picked by a robot
- Placed: the product is placed in a box
- Lost: the product fell off the belt

Transitions

- Tr1: the product is picked by a robot
- Tr2: the product is placed into a box
- Tr3: The product is not being picked by any robot and it has left the work are of the last robot

3.3.2 Boxes and place conveyor model

The boxes are fed to the system with a spacing mechanism so that the distance between boxes is a constant value equals to Δx_b.

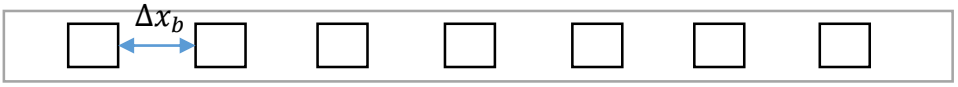

Figure 3-4: Place conveyor

The place conveyor velocity V_{pl} is a manipulated variable. In most cases, however, V_{pl} is computed such that the system is self-balanced. This velocity is computed to match the number of products entering the system with the number of holes per box, as in equation (3.2).

$$V_{pl} = \frac{N_{row}^{p}}{N_{layer}^{b}} \Delta x_b \; PR \qquad \textbf{(3.2)}$$

A box in the system is modelled with a four-state automaton, as shown in Figure 3-5. The states and transitions of the box automaton are described below.

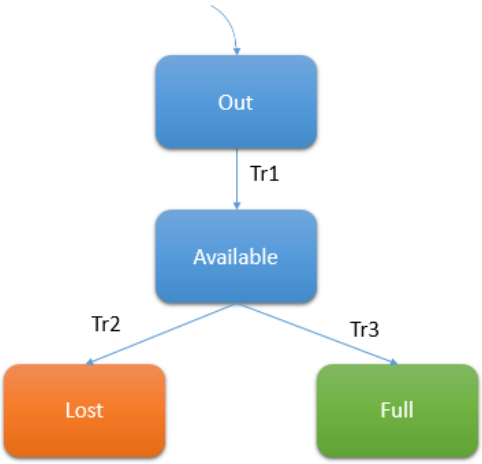

Figure 3-5: Box automaton

States

- Out: the box is not on the place conveyor belt yet
- Available: the box is inside the system and available to receive products
- Full: the number of deposited products equals the number of holes per box
- Lost: the box left the belt missing some products

Transitions

- Tr1: the box enters the system with free layers
- Tr2: the box leaves the place conveyor with some empty layers
- Tr3 the box is filled completely

3.3.3 Robot model

The robot dynamics is described by a four-state timed automaton as shown in Figure 3-6.

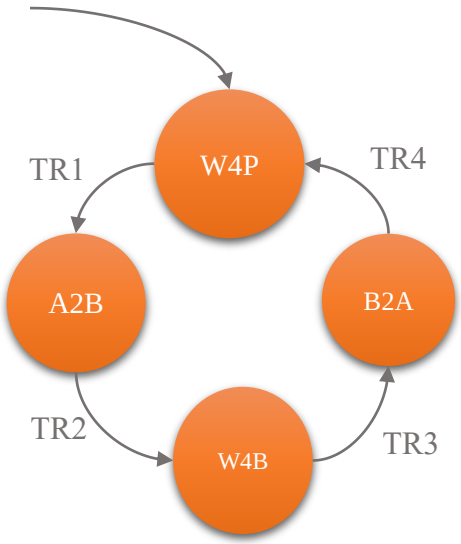

Figure 3-6: Mono grip robot automaton

States

$W4P$: the robot is waiting for a product

$A2B$: the robot is moving from conveyor A to conveyor B

$W4B$: the robot is waiting for a box

$B2A$: the robot is moving from conveyor B to conveyor A

Transitions

TR1: ∃ p: $L^r_{pi,min} \leq x_p \leq L^r_{pi,max}$ **AND** p is assigned to robot r

TR2: A2B state has been active for more than $\frac{T_m}{2}$

TR3: ∃ b: $L^r_{pl,min} \leq x_b \leq L^r_{pl,max}$ **AND** b is assigned to robot r **AND** b is available (not full)

TR4: B2A state has been active for more than $\frac{T_m}{2}$

This automaton models a mono-grip robot. Initially, a robot is in the W4P state where the robot waits for products to enter its working area. Notice that a generic robot picks only products that have been assigned to it by the line manager. In this way, the manager is able to distribute the load among robots and schedule robot tasks. The definition of a suitable scheduling rule (which is the aim of this work) will be discussed in the following chapters. In A2B state, the robot travels from belt A to belt B within one half of its mission time. In W4B state, the robot waits for an assigned box to enter its working area. The robot places products only in available boxes (not completely full). As soon as the product has been deposited, the robot travels back from belt B to A within the other half of the mission time.

This way of modelling lumps the dynamics of the robot into few system parameters which depend on the speed and acceleration of the modelled robots. This model ignores the movement in the z-direction. This is valid assuming the movement along the z-direction is much faster than the travel time from A2B or B2A.

3.4 Model verification

To verify the simplified model, one needs a real data for comparison. Since real data for diverse plant layouts is not readily obtained, the simulation of the behaviour of the plant obtained with the simplified model has been compared with the one obtained with a realistic and reliable simulation environment: Reference Simulation Tool (RST). RST is a simulation environment for packaging lines where all the kinematic constrains and robot trajectories are modelled. The simulator was developed to help plant designers better understand the limitations of different configurations, for instance perpendicular belts. The simulator can also simulate the dynamic behaviour of different robots. This simulator has been tested and verified against numerous real plants which makes it for the purpose of this work the "reality". In other words, if the simplified model behaves similarly to RST then the simplified model is verified.

The verification procedure compares the plant performances of the simplified model with the one simulated with RST. The simulation scenario is the start-up transient, where empty boxes and products start entering the packaging line. The performance indices used to compare the plant behaviour are the number of lost products during the start-up and the time between pickups (the time interval between two consecutive pickups performed by a generic robot). Notice that a generic scheduling rule of products and boxes to robots has to be implemented in order to simulate a realistic scenario and appreciate the system response obtained with the two models. The assignment that is considered here is a cyclic one. A more complete description of this scheduling rule will be later presented. Hence, the simplified model verification procedure (against RTS simulation tool) is summarized as follows:

1- A reference plant is constructed in RST environment
2- A cyclic scheduling strategy to control the plant is implemented
3- The incrementing number of lost products until a constant value is recorded
4- The time between two pickups for each robot is recorded
5- Steps 1 to 4 are repeated with the simplified model
6- The time between two consecutive pickups and the number of lost products on both models is compared

7- The comparison is carried out on different plant layouts

This section describes, first, the cyclic assignment. Second, it highlights the RST model. Third, the section describes the start-up and steady state conditions. Finally, simulation results for validating the simplified model with three different layouts is presented.

3.4.1 Cyclic assignment

Products are assigned to robots with a sequence that is repeating cyclically. This strategy preforms well at the steady state, given that the assumptions in section 3.2 are respected. Considering a plant with three robots, the cyclic assignment of products would be as in Figure 3-7. The figure depicts a top view of a pick conveyor where products are flowing according to the indicated direction. Products are assigned to robots as they pass by the vision system. Similarly, Figure 3-8 depicts the cyclic assignment of layers of boxes. Next, the cyclic assignment is formalized.

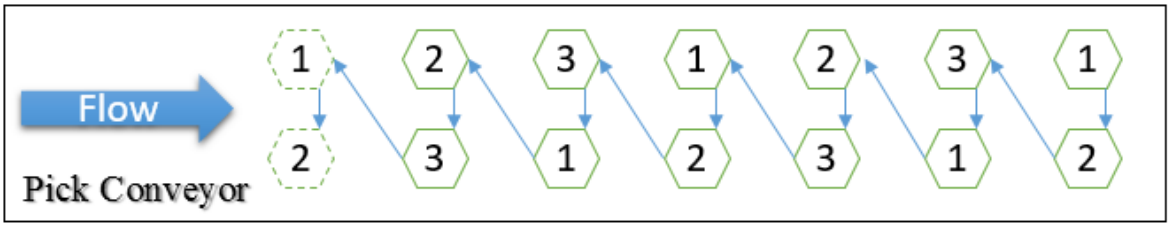

Figure 3-7: Cyclic assignment of products to a team of three robots

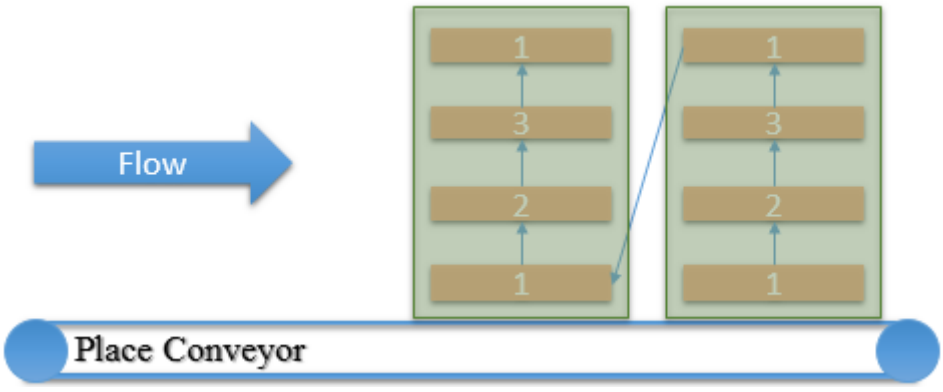

Figure 3-8: Cyclic assignment of boxes to a team of three robots

Given a set of products $\mathcal{P} = \{1,2,..,p,...,P\}$ and a set of robots $\mathcal{R} = [1,2,...r,...,R]$, one form of the cyclic assignment would be defined as the following:

$$S_r = \{p|\ p = nR + r + m\ i - (m-1) * (\mathrm{R} - i)\ \},$$

$$m = \begin{Bmatrix} 1, & i + r \leq \mathrm{R} \\ 0, & i + r > \mathrm{R} \end{Bmatrix}, \forall r \in \mathcal{R},\ n \in \mathbb{N} \cup \{0\}, n \leq \frac{P}{\mathrm{R}}, i \in \{0,1,..,(\mathrm{R}-1)\}$$

Where,

S_r: is the set of product sequence to be packed by robot r

i: is the phase shift of the cyclic assignment

To exemplify, assume that,

$$R = 4, P = 8,$$

Then, the table below shows four sets of products which are scheduled for each robot. Each row of the table shows a shifted version of the cyclic schedule.

Table 2: Cyclic schedule with different phases

Shift by i	S_1	S_2	S_3	S_4
0	{1,5}	{2,6}	{3,7}	{4,8}
1	{2,6}	{3,7}	{4,8}	{1,5}
2	{3,7}	{4,8}	{1,5}	{2,6}

Due to the periodicity of the schedule, there is no need to look into the complete sequence of all products. It is only sufficient to look at the schedule in one period. Consider a cyclic schedule with no phase shift. Products can be arranged in one cyclic as {1, 2, 3, and 4}. The number between the curly brackets are the indices of robots, consult Figure 3-9 for better visualization.

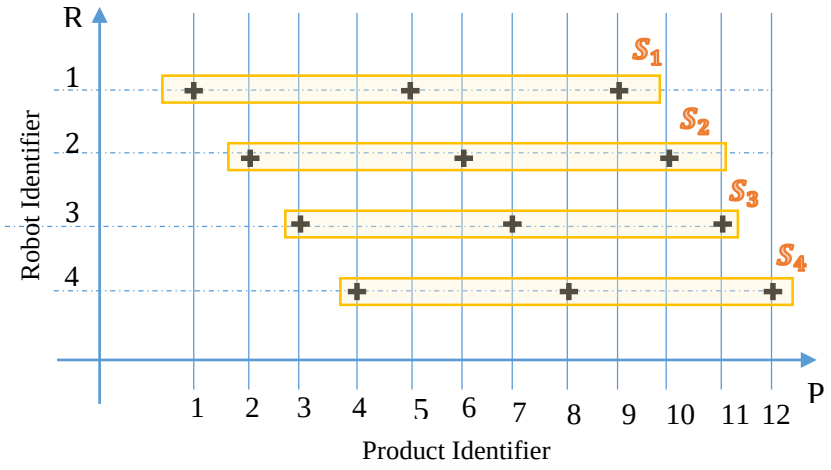

Figure 3-9: A representation of a cyclic assignment with zero shift, a plus sign means that the product in the x-axis is handled by the robot in the y-axis

3.4.2 Start-up transient and steady state conditions

For co-flux plant layouts, a steady state condition is achieved when the products and boxes are assigned to robots such that the number of lost products reaches a constant value. Furthermore, the pick and place positions repeat in a cyclic manner for each robot; this can be

observed by measuring the time between two consecutive pickups. During the start-up transient, the plant experiences an incremental loss of products as well as the robots' behaviours are not cyclic. Figure 3-10 depicts the transient behaviour of a three-robot model plant. During the start-up transient, the number of products reaches 10 while in the steady state the number remains constant. Additionally, it is observed that during the start-up transient the robots are not picking products in a cyclic manner. At the steady state, however, the manipulation of products is cyclic for each robot. Since the number of lost products is the figure of merit that decides the efficacy of the packaging system, the simplified model should reproduce the number of lost products of its real counterpart. Thus, the model should be verified in terms of the number of lost products, refer to sub-sections 3.4.3.1, 3.4.3.2 and 3.4.3.3 for simulation results.

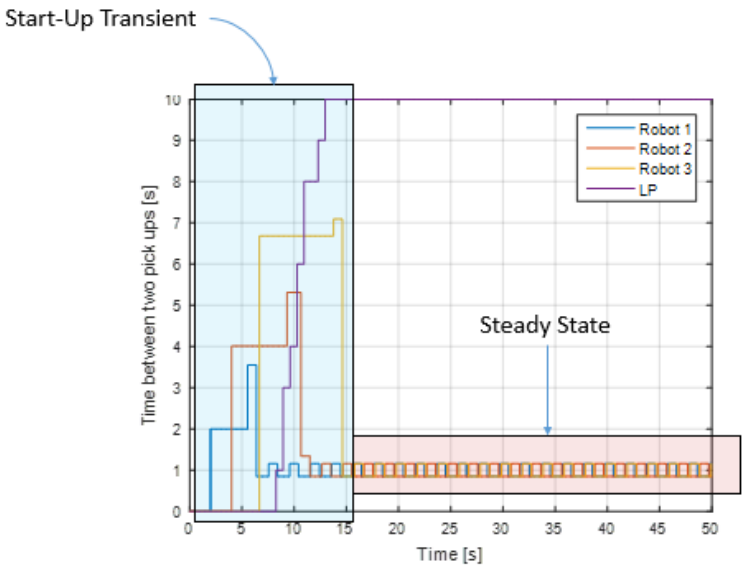

Figure 3-10:Start-up and steady state conditions

3.4.3 Plant layouts

To highlight the significance of a steady state solution, the time between two consecutive pickups is measured for each robot. It is expected that with a cyclic assignment, this time reaches a value equals to the reciprocal of the rate of handling products per robot. The time is calculated according to the following relation:

$$\text{Time Between Two Consecutive Products} = \frac{Robots}{N_{row}^{p} \cdot PR}$$

The sub-sections 3.4.3.1 through 3.4.3.3 describe this behaviour for three different plant layouts.

For the purpose of verifying the simplified model, three different plant layouts were considered. The first two models, 1 and 2, consist of fast acting robots as it is evident from Table 3, the mission time is less than a second. In theory, the system in model 1 can handle up to 216 [ppm], and model 2 can handle up to 289 [ppm]. These numbers are obtained starting from the mission time of 0.83 $\left[\frac{s}{p}\right]$. By inverting this quantity, the product rate per second that can be handled by a robot is obtained. This quantity is then multiplied by the total number robots in the system. Model 3 is a particular case in which a five-grip robot is modelled as a mono grip robot. Therefore, a group of five adjacent products are considered as one large product. A product rate of 600 [ppm] is considered to be 120[gpm], where gpm stands for group per minute. This way of approximating multi-grip robots is discussed later in chapter 8.

Table 3: Plant layouts

MODEL [#]	ROBOTS [#]	N_{row}^{p}	N_{layer}^{b}	PR	V_{pi}	V_{pl}	T_m	Δx_b
1	3	2	2	180	26	13.5	0.83	0.15
2	4	2	2	240	26	18	0.83	0.15
3	6	2	4	120	30	13.5	1.46	0.45

3.4.3.1 Model 1

Figure 3-11 and Figure 3-12 shows a comparison of the behaviour of the simplified model against RST. As said before, the models have been compared during a start-up transient where products and empty boxes start entering the cell. In terms of lost products, both models lose the same number of products at the steady state. The time at which both models start losing products is slightly different. For the time between two pickups, the simplified model shows a cyclic behaviour around one second. This value corresponds to the nominal product rate per robot. This value is usually more than the mission time of robots. Interestingly, RST model

shows a similar behaviour with extra un-modelled dynamics. These results show that the simplified model is reliable.

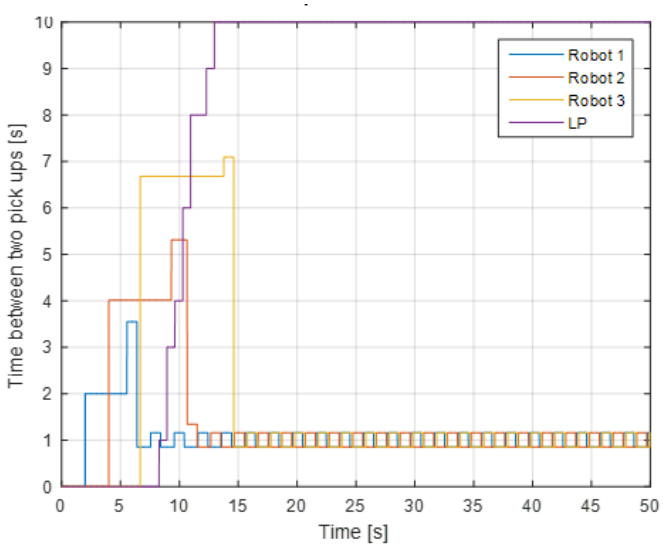

Figure 3-11 Time between two pickups for mono grip model 1 (Simplified Model)

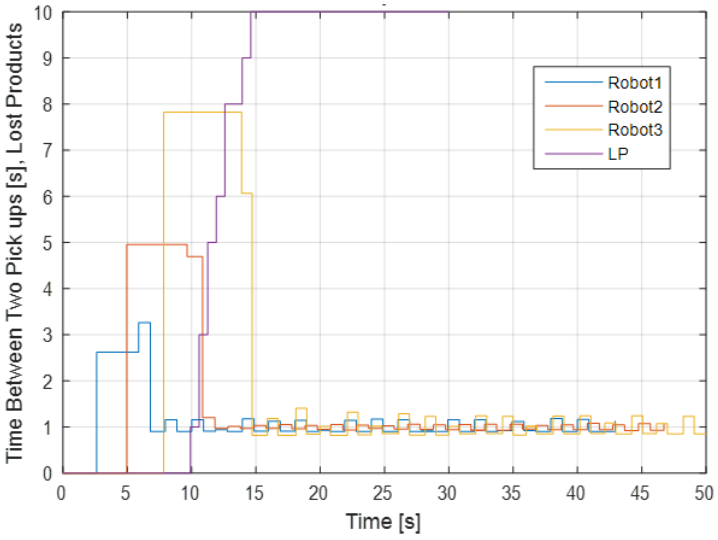

Figure 3-12 Time between two pickups for mono grip model 1 (RST)

3.4.3.2 Model 2

Model 2 is another plant layout over which the comparison is performed. The number of lost products is matched. The initial waiting time is also similar. The simplified model shows a steady state convergence to the mission time of one second. While the RST model shows a cyclic behaviour around the one second value. This again is an acceptable behaviour for the purpose of defining a schedule of the plant.

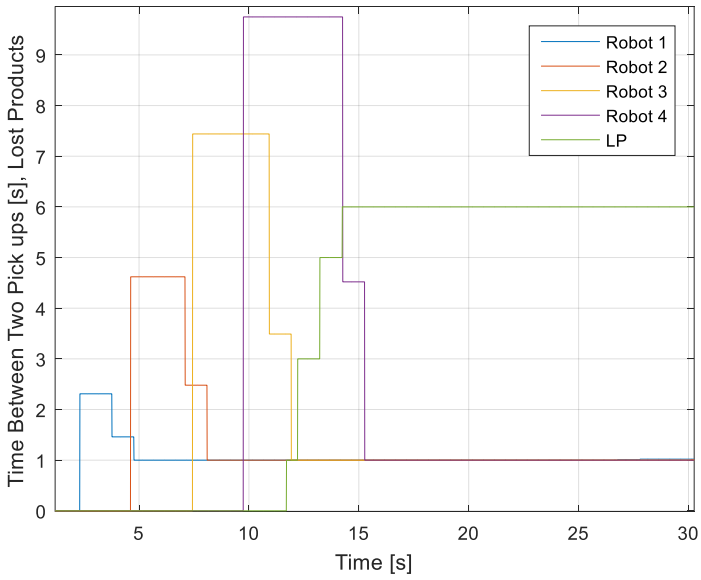

Figure 3-13: Time between two pickups for mono grip model 2 (Simplified Model)

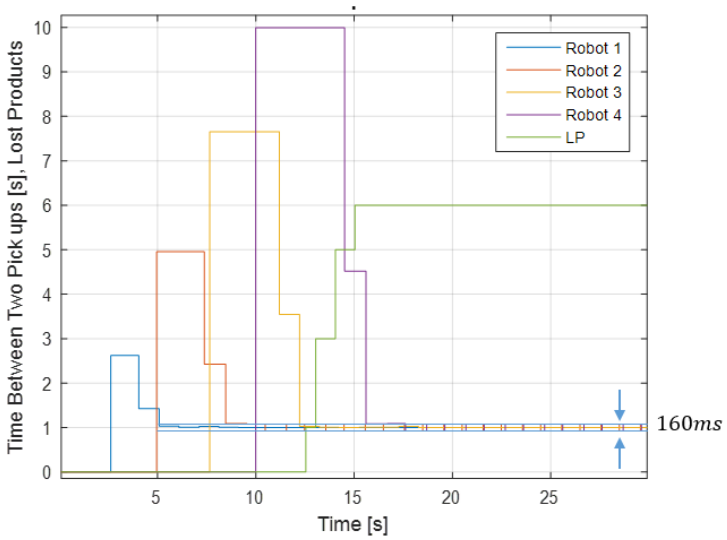

Figure 3-14: Time between two pickups for mono grip model 2 (RST)

3.4.3.3 Model 3

Similar discussion can be stated to this larger plant. The interesting part is the similarity of the steady state behaviour of both models. Once again it is important to note that there is slight mismatch due to un-modelled dynamics, which is not of a major concern when it comes to the assignment problem.

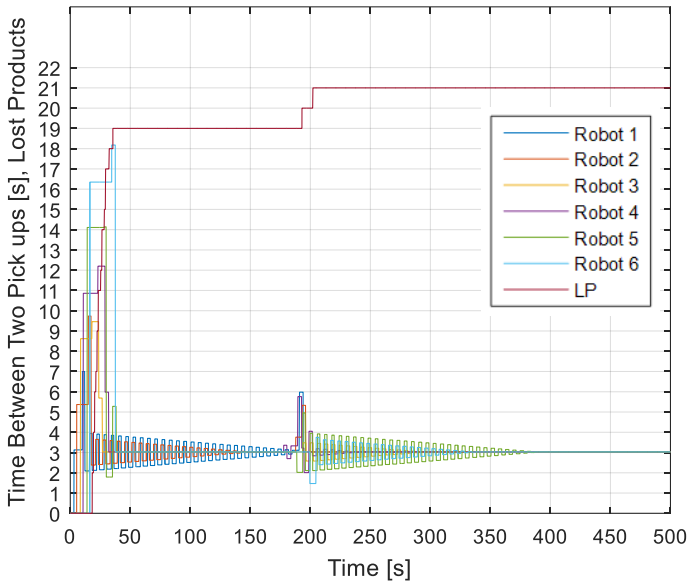

Figure 3-15: Time between two pickups for mono grip model 3 (Simplified Model)

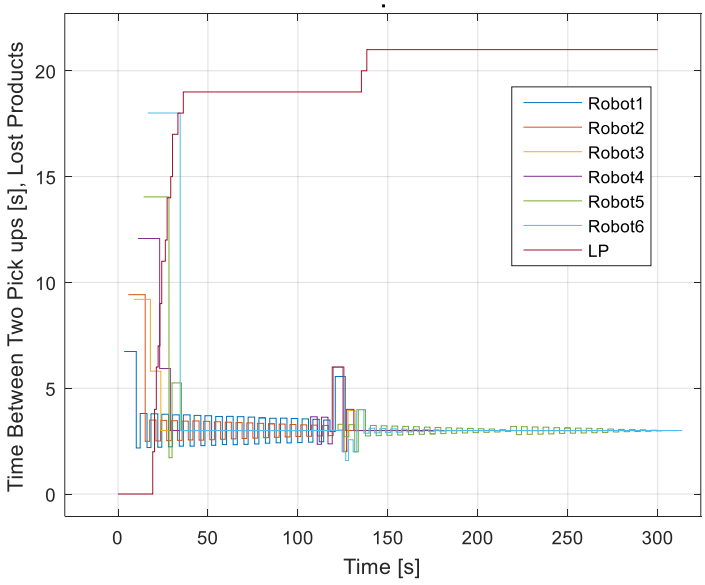

Figure 3-16: Time between two pickups for mono grip model 3 (RST)

3.5 Conclusion

A simplified modelling of the proposed packaging system of two conveyor belts and a team of robots is derived. An intuitive solution to the assignment problem, namely the cyclic algorithm, has been presented. The simplified model has been verified against a more accurate model, RST. The verification is carried out in terms of the total number of lost products, and the time between two consecutive pickups for each robot. Finally, simulation results on three different layouts were plotted to support the reliability of the simplified model. Next chapter presents a model predictive controller for the plant as a solution of a mixed integer linear program.

4 MILP formulation of the assignment problem

4.1 Introduction

To build a schedule of this type of plant, it is decided to study the problem as an optimization problem. In the best case scenario, the controller should be able to assign products to boxes and set the belt velocities to assure minimum loss of products. However, even if the best case scenario was not reached, the effect of changing velocities and different assignment gives a rich knowledge that can be used to develop a heuristic solution.

One of the popular approaches to model difficult problems is the mixed integer linter program (MILP), where some of the decision variables are integer valued and the others are real valued, [1]. The problem is formulated with the two different approaches. One is the future mission approach and the other is based on a network flow problem. Since both solutions are designed to control the plant in terms of setting the future schedule and the belts' velocities, few information need to be fed back to the manager. The controller needs to know the status of robots and the current position of products and boxes on the belts. Moreover, the current belts' velocities, namely (Ω_{pick} and Ω_{place}), needs to be known. Based on these information, the controller gives the future assignment of products and boxes. Due to the continuity of disturbances, the controller needs to update its decision. To do so, the controller is called based on the occurrence of significant events. These events include the reception of new products and boxes, as well as the change of the status of robots. More details in [3].

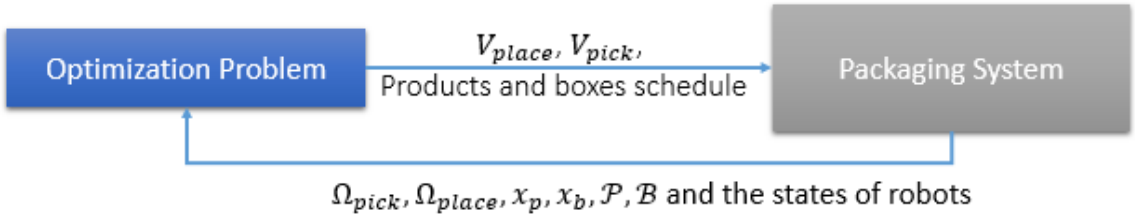

Figure 4-1: Packaging system in a feedback configuration

4.2 Future mission approach

4.2.1 Introduction

Referring to the model in Figure 1-2, each robot has to pick a sequence of products and place them in boxes. The manager needs to develop this sequence respecting the time required to pack one product (the mission time Tm). It also needs to respect work area constraints, and upper and lower bounds of each decision variables. Before jumping ahead into the details, first a description of the decision variables and the problem parameters is presented.

4.2.2 Problem parameters

$\mathcal{P}$ = {1,2,…,P}: set of products on belt A **(4.1)**

$\mathcal{B}$ = {1,2,…,B}: set of boxes on belt B **(4.2)**

$\mathcal{R}$ = {1,2,…,R}: set of robots **(4.3)**

M= min(h_b, P): the maximum number of future missions **(4.4)**

h_b: number of layers per box **(4.5)**

δV: maximum change of velocity **(4.6)**

$[L^r_{pi,min}, L^r_{pi,max}]$: robot work area for picking products **(4.7)**

$[L^r_{pl,min}, L^r_{pl,max}]$: robot work area for placing into boxes **(4.8)**

4.2.3 Decision variables

$\mathbb{a}^r_{m,p}$: 1 if the product p is picked by robot r in mission m

$\alpha^r_{m,p}$: this is a real-valued slack variable equals to $\mathbb{a}^r_{m,p} * w_{pi}$

$\mathbb{b}^r_{m,b}$: 1 if product $\mathbb{a}^r_{m,p}$ is packed into box b by robot r in mission m

$\beta^r_{m,b}$: this is a real-valued slack variable equals to $\mathbb{b}^r_{m,b} * w_{pl}$

$t^r_{pi,m}$: is the picking time for mission m and robot r

$t^r_{pl,m}$: is the placing time for mission m and robot r

w_{pi}: is the reciprocal of Belt A velocity $\frac{1}{V_{pi}}$

w_{pl}: is the reciprocal of Belt B velocity $\frac{1}{V_{pl}}$

In the proposed model of the plant, each robot handles a product in a mission. This mission contains a product, a box, a time to pick and a time to place. Each robot can handle multiple future missions. To model packing products into boxes, Boolean-valued variables are considered. The time and belts' velocities are real-valued variables.

4.2.4 Objective function

The aim of the objective function is to maximize the number of the binary variables which in turn means picking the maximum possible number of products. The MILP solver for which this problem is constructed solves the minimizing problem. Hence, the objective function J is presented as the following.

$$\min J = \sum_{r\in\mathcal{R}}\sum_{m\in\mathcal{M}}\left(\rho^T\cdot\left(t^r_{pi,m}+t^r_{pl,m}\right)+\sum_{p\in\mathcal{P}}\boldsymbol{\Gamma}^{\boldsymbol{r}}_{\boldsymbol{m,p}}\cdot\mathbb{a}^r_{m,p}+\sum_{b\in\mathcal{B}}\boldsymbol{\Psi}^{\boldsymbol{r}}_{\boldsymbol{m,b}}\cdot\mathbb{b}^r_{m,b}\right) \quad (4.9)$$

Where,

$$\Gamma^{r}_{m,p} = \frac{\Gamma^{min}}{v}x_p - \frac{\Gamma^{min}}{v}L^r_{pi,max} + \Gamma^{min} \qquad x_p < L^r_{pi,max} \quad (4.10)$$

$$\Gamma^{r}_{m,p} = -\Gamma^{min} \qquad x_p \geq L^r_{pi,max} \quad (4.11)$$

$$\Gamma^{min} < 0 \quad (4.12)$$

$$\boldsymbol{\Psi}^{\boldsymbol{r}}_{\boldsymbol{m,b}} = \frac{\Gamma^{min}}{v}x_b - \frac{\Gamma^{min}}{v}L^r_{pi,max} + \Gamma^{min} \qquad x_b < L^r_{pl,max} \quad (4.13)$$

$$\boldsymbol{\Psi}^{\boldsymbol{r}}_{\boldsymbol{m,b}} = -\Gamma^{min} \qquad x_b \geq L^r_{pl,max} \quad (4.14)$$

$$\rho^T > 0 \quad (4.15)$$

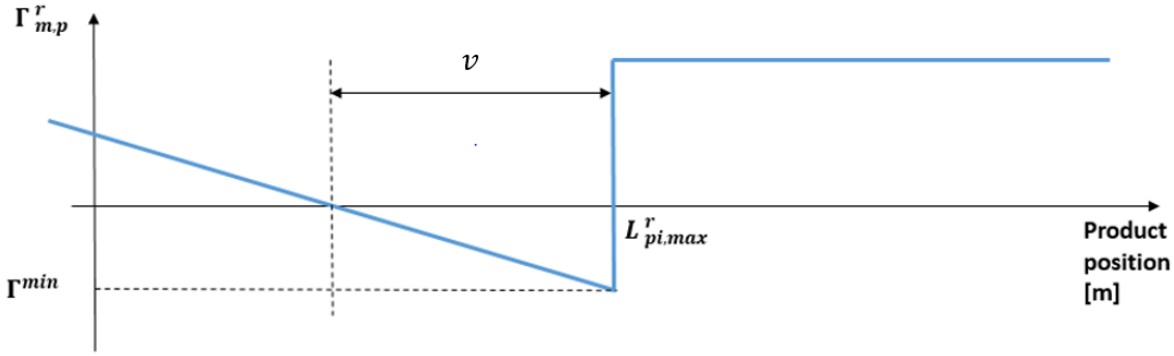

Figure 4-2: Linear weighting decay

One notices that the weights on the Booleans are decreasing as the products and boxes approach the maximum limits of robots. This means that the downstream products and boxes are preferred to the ones on the upstream, Figure 4-2 shows the graph of the weight as a function of the position. There is also an important parameter which is referred to as v and stands for the visibly region. This parameter indicates how far in the future the solver could look, so the higher the v the more products will be considered. The positive weight on the picking and placing time is to force the solution to pack products into boxes in the shortest possible time.

4.2.5 Problem constraints

The problem constraints are divided into three main groups. One is the physical constraints which has to be respected in order for the solver to give a feasible solution to the

plant in terms of the maximum number of products and boxes in a mission, work areas, mission time and velocities. The limits for these quantities come from the simplified model. The second set of constraints deal with obtaining equivalent linear representation of non-linear constraints. The final set of constraints deal with the mission-in-progress constraints which requires that if a product is held by a robot the solution should respect that and construct the solution based on this state. This section describes the constraints to a certain level of details.

4.2.5.1 Work area constraints

Each robot has a work area where it can pick a product and another area for placing into a box. The solution should respect these areas. Equations (4.16) through (4.19) are the linear constrains that respect the lower and upper bounds of the work areas.

$$L^r_{pi,min} \cdot w_{pi} - t^r_{pi,m} - \sum_p \alpha^r_{m,p} \cdot x^P_p + \sum_p \mathbb{a}^r_{m,p} \cdot U \leq U \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.16)}$$

$$-L^r_{pi,max} \cdot w_{pi} + t^r_{pi,m} + \sum_p \alpha^r_{m,p} \cdot x_p + \sum_p \mathbb{a}^r_{m,p} \cdot U \leq U \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.17)}$$

$$L^r_{pl,min} \cdot w_{pl} - t^r_{pl,m} - \sum_b \beta^r_{m,b} \cdot x_b + \sum_b \mathbb{b}^r_{m,b} \cdot U \leq U \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.18)}$$

$$-L^r_{pl,max} \cdot w_{pl} + t^r_{pl,m} + \sum_b \beta^r_{m,b} \cdot x_b + \sum_b \mathbb{b}^r_{m,b} \cdot U \leq U \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.19)}$$

To highlight the idea, one of these constrains are considered. For instance (4.16), where the lower bound of the work area of robots on belt A is respected. It can be noted that U is a large constant value utilized in order to satisfy the constraint even when the sum of the binary values $\sum_p \mathbb{a}^r_{m,p}$ is zero. In this condition, equation (4.16) becomes:

$$L^r_{pi,min} \cdot w_{pi} \leq U + t^r_{pi,m}$$

On the contrary, if the sum of the same Boolean-values is one, the constraint becomes:

$$(L^r_{pi,min} - x^P_p) w_{pi} \leq t^r_{pi,m}$$

Which means that the pick time for that product in the m^{th} mission needs to satisfy the minimum area constraint. The same discussion applies to equations (4.17) through (4.19).

4.2.5.2 Time continuity constraints

One of these sets of constraints corresponds to respecting a time equal to $\frac{T_m}{2}$ between the picking and placing actions in a mission. And the other set corresponds to respecting a time equal to $\frac{T_m}{2}$ between the placing action in a mission and the picking action in the next mission.

$$t^r_{pi,m} + \frac{T_m}{2} \leq t^r_{pl,m} \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.20)}$$

$$t^r_{pl,m} + \frac{T_m}{2} \leq t^r_{pi,m+1} \qquad m \in \mathcal{M} - 1, r \in \mathcal{R}, \quad \textbf{(4.21)}$$

4.2.5.3 Special order sets

Recall that a Boolean-valued set of decision variables is assumed in order to model the picking and placing action. Consequently, there exists in every mission for every robot a variable corresponding to a product p. Therefore, a set of constraints needs be added in order to constraint the solver to find only one Boolean-valued number corresponding to a product. This is described by constraints (4.22) and (4.23). Same consideration apply to the set of boxes except for the fact that a box can hold multiple products. Hence, the total number of Boolean-valued variables corresponding to a box needs to be equal to the number of available places in that box, constraint (4.25). Constraints (4.23) and (4.24) are similar and serve as to match a product for a box in a mission m for a robot r.

$$\sum_r \sum_m \mathbb{a}^r_{m,p} \leq 1 \qquad p \in \mathcal{P}, \quad \textbf{(4.22)}$$

$$\sum_p \mathbb{a}^r_{m,p} \leq 1 \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.23)}$$

$$\sum_b \mathbb{b}^r_{m,b} \leq 1 \qquad m \in \mathcal{M}, r \in \mathcal{R}, \quad \textbf{(4.24)}$$

$$\sum_r \sum_m \mathbb{b}^r_{m,b} \leq h_b \qquad b \in \mathcal{B}, \quad \textbf{(4.25)}$$

4.2.5.4 Maximum velocity change constraint

The following constraints are introduced in order to avoid rapid and uncontrolled changes of the conveyor velocities. The conveyor velocity profile has to be smooth to avoid the breaking of mechanical components. In this way the difference between the new velocity and the actual one is reduced below certain limits.

$$\left|V_{pi}(k) - V^{prev}_{pi}\right| \leq \delta V \quad \textbf{(4.26)}$$

$$\left|V_{pl}(k) - V^{prev}_{pl}\right| \leq \delta V \quad \textbf{(4.27)}$$

4.2.5.5 Multiplication constraints

These sets would not have been necessary if the belts' velocities were assumed to be constant. To demonstrate this, consider the work area constraint (4.16). For constant velocities the constraint is written as the following:

$$L^r_{pi,min} \leq V_{pi} t^r_{pi,m} + \sum_p \mathbb{a}^r_{m,p} \cdot x^P_p$$

But if V_{pi} is a decision variable, the term $V_{pi} t^r_{pi,m}$ is non-linear and needs be changed to a linear constraint. This can be accomplished by adding a set of linear constraints that is equivalent to

$V_{pi}t^r_{pi,m}$. To get to equation (4.16) multiply by $\omega_{pi} = \frac{1}{V_{pi}}$. The term to treat becomes $w_{pi}\mathbb{a}^r_{m,p}$. Now add a slack variable $\alpha^r_{m,p} = w_{pi}\mathbb{a}^r_{m,p}$, the result of the products is as the following:

IF $\mathbb{a}^r_{m,p} = 0$ **OR** $w_{pi} = 0$ **THEN** $\alpha^r_{m,p} = 0$

AND IF $\mathbb{a}^r_{m,p} = 1$ **THEN** $\alpha^r_{m,p} = w_{pi}$

The constraints (4.28) through (4.31) satisfy the previous and hence the multiplication is transferred into a set of linear constraints. The same applies to the boxes, constraints (4.32) through (4.35).

$$-\alpha^r_{m,p} + w_{pi} + M^{pi} \cdot \mathbb{a}^r_{m,p} \leq M^{pi} \qquad p \in \mathcal{P}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.28)$$

$$\alpha^r_{m,p} - M^{pi} \cdot \mathbb{a}^r_{m,p} \leq 0 \qquad p \in \mathcal{P}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.29)$$

$$-\alpha^r_{m,p} \leq 0 \qquad p \in \mathcal{P}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.30)$$

$$\alpha^r_{m,p} - w_{pi} \leq 0 \qquad p \in \mathcal{P}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.31)$$

$$-\beta^r_{m,b} + w_{pl} + M^{pl} \cdot \mathbb{b}^r_{m,b} \leq M^{pl} \qquad b \in \mathcal{B}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.32)$$

$$\beta^r_{m,b} - M^{pl} \cdot \mathbb{b}^r_{m,b} \leq 0 \qquad b \in \mathcal{B}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.33)$$

$$-\beta^r_{m,b} \leq 0 \qquad b \in \mathcal{B}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.34)$$

$$\beta^r_{m,b} - w_{pl} \leq 0 \qquad b \in \mathcal{B}, m \in \mathcal{M}, r \in \mathcal{R} \qquad (4.35)$$

4.2.5.6 Mission in progress constraints

These constraints describe the situation when a robot r_0 has picked a product p_0. Therefore, the robot is in A2B or W4B states. The decision variable related to r_0 and p_0 should be set to one in the first mission. This way the solution finds other products to fill the remaining missions. When robot r_0 has just picked a product a timer $\bar{t}$ is set such that $\bar{t} := 0$. This timer keeps the information about the time the robot is busy. Constraint (4.20) imposes that the place time must always be $\frac{Tm}{2}$ after the picking time. However, p_0 has already been picked and once r_0 has travelled to belt B, there is no need to wait $\frac{Tm}{2}$ to pack the product, hence, constraint (4.37) is added. Constraint (4.36) fixes p_0 to the first mission.

$$\mathbb{a}^{r_0}_{1,p_0} = 1 \qquad p_0, r_0 \in WIP \qquad (4.36)$$

$$t^{r_0}_{pi,1} = -\min(\bar{t}, \tfrac{Tm}{2}) \qquad r_0 \in WIP, \qquad (4.37)$$

When the robot has placed a product, the robot is in the B2A state. In this state, constraint (4.36) is removed for robot r_0 and mission 1.The pick time for the second mission time is constrained according to constraint (4.38).

$$\frac{T'_m}{2} \leq t^{pick}_{r_0,2}, \text{ where } T'_m = 2 * \max(\bar{t}, 0) \qquad r_0 \in WIP \qquad \textbf{(4.38)}$$

Where $\bar{t}$ is the time calculated from placing a product by the robot r_0 in B2A state until the solver is called.

4.2.5.7 Products and boxes balance constraint

$$\sum_p \mathbb{a}^r_{m,p} - \sum_b \mathbb{b}^r_{m,b} = 0 \qquad m \in \mathcal{M}, r \in \mathcal{R}, \qquad \textbf{(4.39)}$$

4.2.5.8 Upper and lower bounds

$$\mathbb{a}^r_{m,p} \in \{0,1\} \qquad b \in \mathcal{B}, m \in \mathcal{M}, r \in \mathcal{R} \qquad \textbf{(4.40)}$$

$$\mathbb{b}^r_{m,b} \in \{0,1\} \qquad b \in \mathcal{B}, m \in \mathcal{M}, r \in \mathcal{R} \qquad \textbf{(4.41)}$$

$$-w_{pi} \leq 0 \qquad \textbf{(4.42)}$$

$$-w_{pl} \leq 0 \qquad \textbf{(4.43)}$$

4.2.6 Simulations and results

This formulation has been tested on model 1. The main objective of the scheduler is to pack the maximum number of products during start-up transient. However, there are other important results that come in conjunction with the main objective. First, the solution is an online optimizer which means that computational cost is what defines the feasibility of online implementation. It is also known that there are other solutions that can reach the main objective, so observing the final assignment is crucial. Finally, the velocity profile is set by the problem solution and such a profile is plotted as well.

For an online controller to be realized on a real application, it needs to find a solution under a fraction of a second. Otherwise, the decision would be too late. For this reason, this solution is less likely to be practical. Figure 4-4 shows the cost of the solver plotted on the y-axis versus the time instance at which the solver is called. For a simulation of 40[s], the solver's total cost is around an hour and a half. There are few workarounds to make the solution faster, however, the optimality is compromised. To reduce the computational time, one could reduce the prediction horizon. Another solution would be not to consider all the products on the belt, but rather considering only the products that reaches the work areas of the manipulating robots (i.e. $\{p | x_p \geq L^{pick}_{1,min}\}$). The computational time can be reduced considering a lower number of products in the optimization problem. In this section the comparison between the proposed formulation (I-OLM) and a computationally improved solution is proposed (CI-OLM).

- **Nominal solution (I-OLM):** all the products from the vision system until the end of the conveyor are considered.
- **Computationally improved solution (CI-OLM):** the products from the vision system up to the beginning of the work area of the first robot are not considered in the optimization problem (see the dark area in Figure 4-3).

Figure 4-3: CI-OLM

It is also observed that considering all boxes such that $x_b \geq L_{1,min}^{place}$ is not a good solution. This is partially due to the fact that V_{pl} is a manipulated variable, and since every box on the belt has the same velocity, considering all boxes essentially results in a better solution.

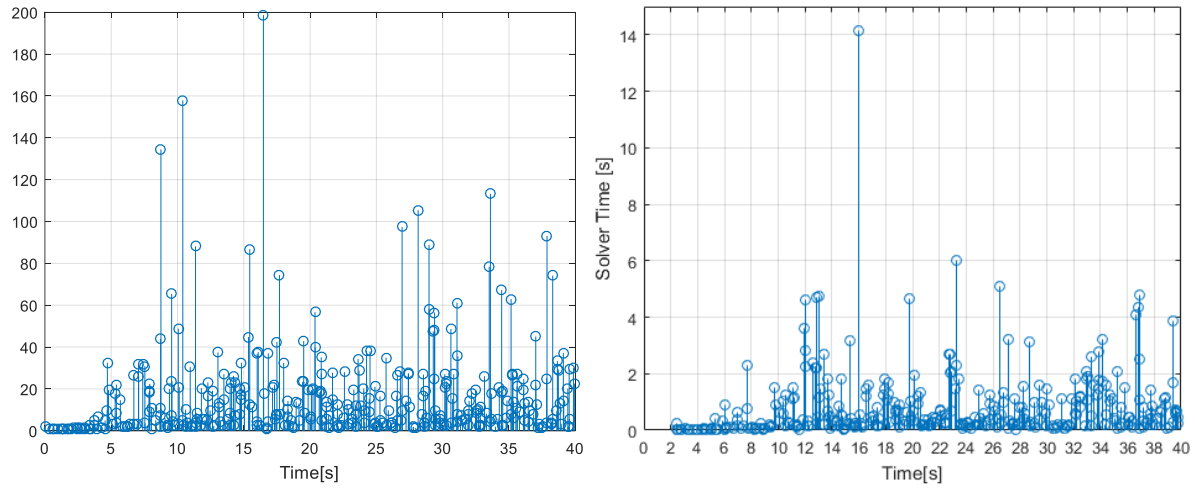

Figure 4-4: Computational cost of I-OLM (left), Computational cost of CI-OLM (right)

Regarding the velocity profile, the main solution and the computationally improved one result in a similar behaviour of, initially, trying to catch up with products then slow down to balance out the plant. However, the dynamic change of the profile is quite different. Figure 4-5 on the left, shows the velocity decision when all products on the belt are considered. The figure on the right is the decision based on products within the work areas of robots. It is interesting to note that even though V_{pl} is only decided based on the previous velocity, the velocity profile tends to reach the expected V_{pl}. In fact, this profile inspired what later is referred to as optimizing over IBD in chapter **Error! Reference source not found.**.

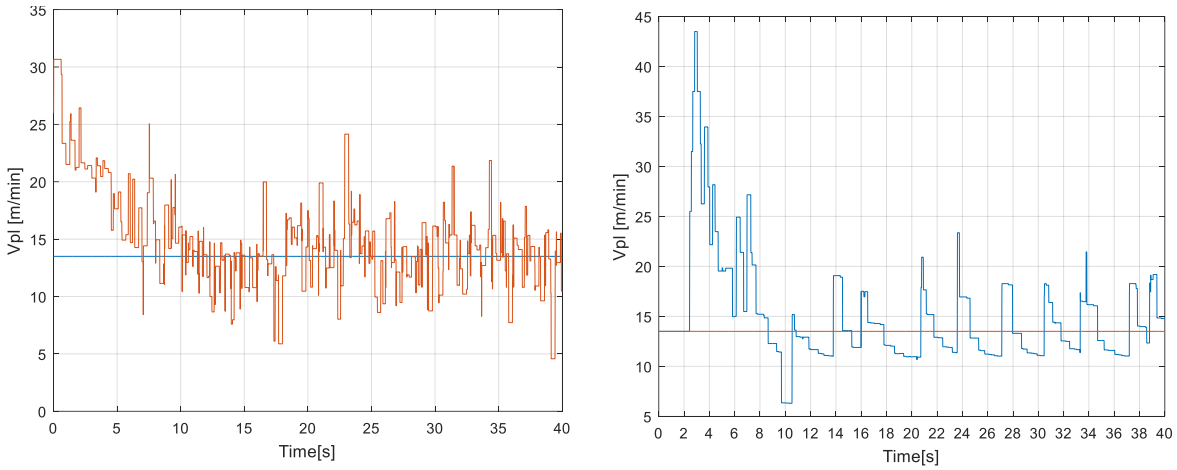

Figure 4-5: Vpl profile considering I-OLM (left) and CI-OLM (right)

It is true that the solution is not practical in terms of computational cost. Nonetheless, the assignment result can still be applied. In other words, the idea is to simulate the problem using the simplified model and record the assignment vector. Once the simulation is done, the assignment vector is applied to the real plant.

Figure 4-6 and Figure 4-7 depict the assignment vector. Although the two vectors are different, both solutions tends to assign more products to robots 1 and 2. For the first 100 packed products, Table 4 tabulates the product distribution among robots.

Table 4: Prodcut distribution among robots

Robot ID	I-OLM	CI-OLM
1	40	41
2	36	39
3	24	20

This distribution can also be observed from Figure 4-8 where the time between two consecutive pickups of robots is measured. Robots 1 and 2 show less and quasi uniform time compared to robot 3.

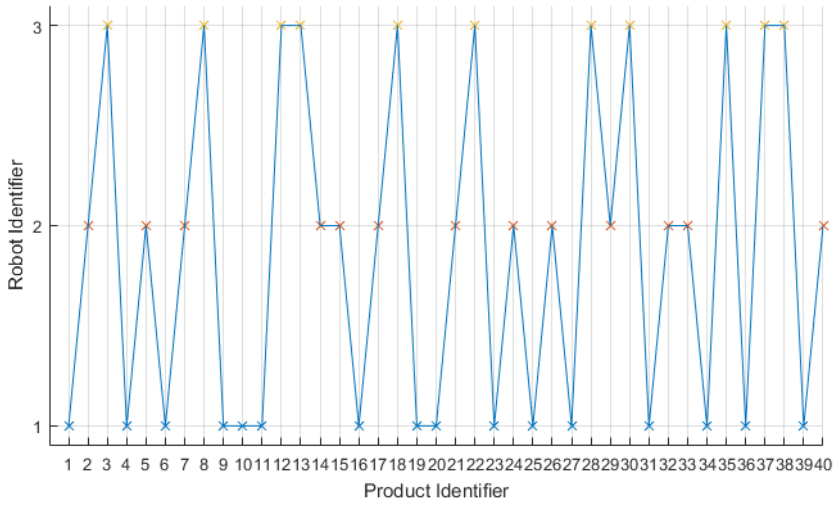

Figure 4-6: Products' assignment: considering I-OLM

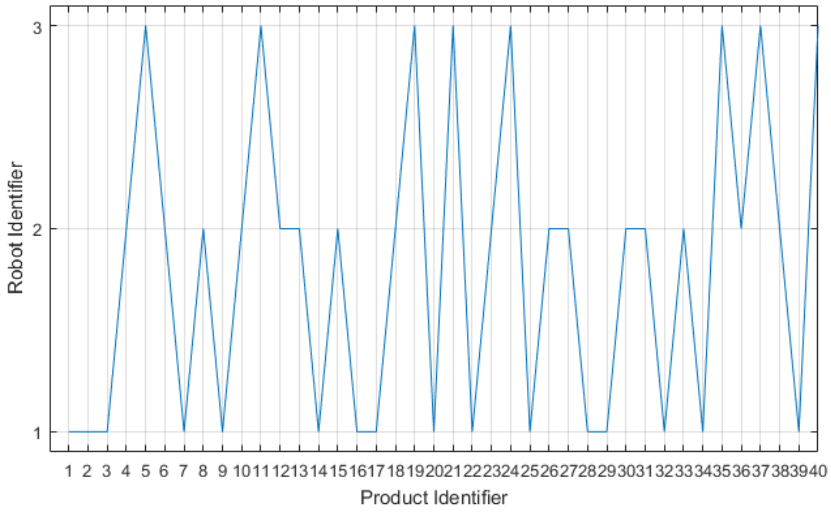

Figure 4-7: Products' assignment: considering CI-OLM solution

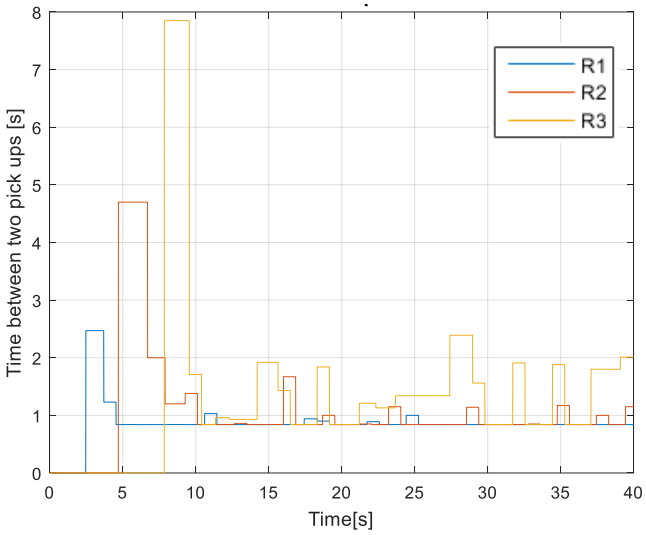

Figure 4-8: Time between two pickups

The number of lost products in the start-up transient is successfully reduced which shows a superiority of this modelling approach over the cyclic assignment. The two proposed solutions reduced the number of lost products, Table 5.

Table 5: Comparing scheduler performance in terms of the number of lost products

Control technique	**Lost prod.**
Optimal Line Manager – Future mission (I-OLM)	0
Optimal Line Manager – Future mission (CI-OLM)	0
Cyclic assignment	9

4.2.7 Conclusion

The scheduling problem has been formulated as a mixed integer linear programing optimization. This solution solves the start-up transient problem. Nonetheless, implementing this on a real system is not practical due to the computational cost. It is worth noting that the controller tries initially to increase V_{pl} and reaches later the expected value to balance the system. By observing the time between two pickups for each robot, one can spot that the load is not uniformly distributed among them. In fact, the first two robots handle most of the load. This solution appears as considering the first two robots greedy. Next in chapter 5, a different formulation has been presented in order to support the previous results.

5 Network flow approach

5.1 Introduction

This modelling approach is different from the previous one in terms of how the future missions are handled. In this approach, products are considered as nodes in a network and the aim of the scheduler is to find the best path to collect the maximum number of products. Suitable constraints have to be defined in order to ensure that downstream products are handled before upstream ones. For each robot, a path is defined and no crossing of paths is allowed.

The arcs of the network connects products with each other and corresponds to the picking action of two products, sequentially. The tail end of an arc is referred to as p and the head end is referred to as q. Indices p and q refer to products and have to be defined in order to ensure that $x_p^{pick} > x_q^{pick}\ \forall(p < q)$, the arcs are drawn in Figure 5-1. In addition, a sink and a source nodes must be defined. These two nodes are needed for the continuity constraint of the network flow and should in no way enter the physical constraints of the system. In this formulation, these nodes are referred to as dummy source and sink products.

Figure 5-1 shows a reference model of three robots, two rows of products and boxes with two layers. As shown in the figure, the two dummy products have no physical presence on the belt. Moreover, all paths must be sourced from the dummy source product and sunk into the dummy sink product. In the figure, the sourcing node is always the one to be packed, and the sinking node is the one that needs to be packed immediately after. In the case of the dummy source product, the dummy product will be packed into a dummy box b equals to zero.

To reflect this problem, a suitable set of parameters are listed, first. Then, a suitable definition of the decision variables are presented. In addition, an objective function subject to a number of constraints is defined. Finally, the problem is constructed mathematically and then numerical results are highlighted.

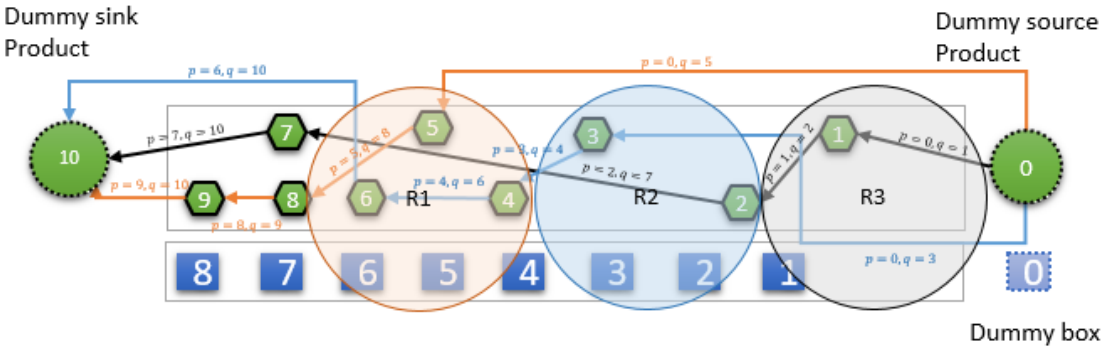

Figure 5-1: Three robot plant model

5.2 Problem parameters

This is the list of parameters that reflects the plant model in Figure 5-1.

$\mathcal{P}$ = {1,2,…,P}: set of products on belt A **(5.1)**

$\mathcal{B}$ = {1,2,…,B}: set of boxes on belt B **(5.2)**

$\mathcal{R}$ = {1,2,…,R}: set of robots **(5.3)**

N_{layer}^{b}: Number of layers per box **(5.4)**

$[L_{pi,min}^{r}, L_{pi,max}^{r}]$: robot work area for picking products **(5.5)**

$[L_{pl,min}^{r}, L_{pl,max}^{r}]$: robot work area for placing into boxes **(5.6)**

$\mathcal{P}_r = \{p \in \mathcal{P} \mid x_p^{pick} < L_{pi,max}^{r}\}$: set of products that can be scheduled for robot r **(5.7)**

$\mathcal{P}'_r = \mathcal{P}_r \cup \{\mathrm{P}+1\}$: set of products that can be scheduled for robot r **(5.8)**

$\mathcal{B}_r = \{b \in \mathcal{B} \mid x_b^{place} < L_{pl,max}^{r}\}$: set of boxes that can be scheduled for robot r **(5.9)**

$\underline{L}_{rp}^{pick} = \max(L_{pi,min}^{r} - x_p^{pick}, 0)$: the relative distance of a product p to minimum distance of robot r **(5.10)**

$\overline{L}_{rp}^{pick} = L_{pi,max}^{r} - x_p^{pick}$: the relative distance of product p to the max distance of robot r **(5.11)**

$\underline{L}_{rb}^{place} = \max(L_{pl,min}^{r} - x_b^{place}, 0)$: the relative distance of a box b to minimum distance of robot r **(5.12)**

$\overline{L}_{rb}^{place} = L_{pl,max}^{r} - x_b^{place}$: the relative distance of box b to the max distance of robot r **(5.13)**

$\left[\underline{V}^{pick}, \overline{V}^{pick}\right]$: velocity limits of belt A **(5.14)**

$\left[\underline{V}^{place}, \overline{V}^{place}\right]$: velocity limits of belt B **(5.15)**

$V_{min}^{pick} = \max(\Omega^{pick} - \delta V, \underline{V}^{pick})$: lower velocity bound for belt A **(5.16)**

$V_{max}^{pick} = \min(\Omega^{pick} + \delta V, \overline{V}^{pick})$: upper velocity bound for belt A **(5.17)**

$V_{min}^{place} = \max(\Omega^{place} - \delta V, \underline{V}^{place})$: lower velocity bound for belt B **(5.18)**

$V_{max}^{place} = \min(\Omega^{pick} + \delta V, \overline{V}^{place})$: upper velocity bound for belt B **(5.19)**

$Q_{rp} = \overline{L}_{rp}^{pick} \frac{V_{max}^{place}}{V_{min}^{pick}} + \frac{T_M}{2} V_{max}^{place}$: upper bound for constraint (5.30) **(5.20)**

$$R_{rp} = \max\left\{\frac{T_M}{2} V_{max}^{place} - \underline{L}_{rp}^{pick} \frac{V_{min}^{place}}{V_{max}^{pick}}, 0\right\}: \text{lower bound for constraint (5.31)} \quad \textbf{(5.21)}$$

5.3 Decision variables

The decision variables are defined as:

e_{rpqb}: 1 if the product p is packed into box b by robot r and then product q ($> p$) immediately follows (arcs from p to q for each robot)

e_{r0p0}: 1 if the product p is the first to be packed by robot r (arcs from p =0 to p)

vt_{rp}^{pick}: The pickup time of product p for robot r multiplied by v^{place}

v^{place}: Velocity of the place conveyor

$\omega = \frac{v^{place}}{v^{pick}}$: The ratio of the velocity of the place conveyor to that of the pick conveyor

5.4 Objective function

$$\min \sum_{b \in \mathcal{B}_r} \rho_b \left(N_{layer}^b - \sum_{\substack{r \in \mathcal{R} \\ \mathcal{B}_r \ni b}} \sum_{\substack{p,q \in \mathcal{P}_r \\ p<q}} e_{rpqb} \right) - \omega_1 v^{place} + \omega_2 \omega \quad \textbf{(5.22)}$$

The goal of the objective function (5.22) is to accelerate both conveyors while packing the maximum number of products. The weight (ρ_b) is defined as to give more weight on downstream boxes, as expressed by (5.23). Moreover, the weight ρ_b is set much larger than ω_1 and ω_2 in order to prefer the decision of packing the largest number of products over changing the velocities.

$$\rho_b = C + b \qquad C > 0, |C| \gg |\omega_1|, |\omega_2|, \; b \in \mathcal{B}_r' \quad \textbf{(5.23)}$$

5.5 Problem constraints

Figure 5-1 plots a feasible solution where each robot is assigned a path, through which a sequence of products are to be packed. The path is generated such that the product sequence p and q are to be packed when the products enter the work area of a robot. As a result, the controller would assign a suitable packing time for products p and q, the packing "times" are referred to as vt_{rp}^{pick} and vt_{rq}^{pick}. These two variables have a displacement unit of measure and, therefore, could be considered as the "packing positions". Representing the packing action with these decision variables reduced the complexity of the problem constraint definition. In comparison with the previous formulation, this formulation has less number of constraints. In the subsequent sections, the problem constraints are presented in details.

5.5.1 Special order sets

These sets of constraints limit the maximum number of arcs that include products and boxes. Constraint (5.24) assures that a product is assigned only once to a robot r. Similarly, constraint (5.25) assures that the maximum number of products assigned to box b is lower than or equal to N_{layer}^{b}.

$$\sum_{\substack{r\in\mathcal{R}\\ \mathcal{P}_r\ni p}}\sum_{\substack{q\in\mathcal{P}_r'\\ q>p}}\sum_{b\in\mathcal{B}_r} e_{rpqb} \le 1 \qquad p\in\mathcal{P}, \qquad \textbf{(5.24)}$$

$$\sum_{\substack{r\in\mathcal{R}\\ \mathcal{B}_r\ni b}}\sum_{\substack{p,q\in\mathcal{P}_r'\\ p<q}} e_{rpqb} \le N_{layer}^{b} \qquad b\in\mathcal{B}, \qquad \textbf{(5.25)}$$

5.5.2 Network flow constraints

Constraint (5.27) assures that each node on the network has only one incoming arc and one outgoing arc. Which means that when an arc starts from a product p and ends at a product q for a robot r, the following arc for the same robot should start from product q. This flow continues until the final arc is sunk into the dummy sink node. Constraint (5.26) assures that every flow is generated from the dummy source node. The number of sourced arcs from this node is not limited.

$$\sum_{p\in\mathcal{P}_r'} e_{r0p0} = 1 \qquad r\in\mathcal{R}, \qquad \textbf{(5.26)}$$

$$e_{r0p0} + \sum_{\substack{s\in\mathcal{P}_r\\ s<p}}\sum_{b\in\mathcal{B}_r} e_{rspb} = \sum_{\substack{q\in\mathcal{P}_r'\\ q>p}}\sum_{b\in\mathcal{B}_r} e_{rpqb} \qquad p\in\mathcal{P}_r, r\in\mathcal{R} \qquad \textbf{(5.27)}$$

5.5.3 Time window constraint

Recall that ω is the ratio of the belts' velocities $\frac{v^{place}}{v^{pick}}$ and the ratio $\frac{\underline{L}_{rp}^{pick}}{v^{pick}}$ is the time required for product p to reach the minimum bound of the work area of robot r. Similarly, the ratio $\frac{\overline{L}_{rp}^{pick}}{v^{pick}}$ is the time required for a product p to reach the maximum bound of the work area of robot r. Thus, the pickup time must be such that the product p is within the work area of a robot r. This is assured by constraining v^{place}, ω and vt_{rp}^{pick} by constraint (5.28).

$$\underline{L}_{rp}^{pick}\omega \le vt_{rp}^{pick} \le \overline{L}_{rp}^{pick}\omega \qquad p\in\mathcal{P}, r\in\mathcal{R}, \qquad \textbf{(5.28)}$$

5.5.4 Precedence constraints

For better interpretation of these sets, the constraints are divided by v^{place}. Constraint (5.29) assures a minimum of T_m seconds between packing the sequence of products p and q.

The term $T_m V_{max}^{place}\left(\sum_b e_{rpqb} - 1\right)$ is added to assure that this constraint is satisfied even when a product is not picked. Constraint (5.30) assures that the placing time, which is the sum of the pick time plus $\frac{T_m}{2}$, must be less than the time after which a box b reaches the maximum work area of robot r. The term Q_{rp}, as computed by (5.20), is an upper bound added to assure the satisfaction of the constraint, if e_{rpqb} is equal to zero. Constraint (5.31) assures a minimum of $\frac{Tm}{2}$ between the last place time and the next pick time. Similar to Q_{rp}, R_{rp}, as expressed by (5.21), is a lower bound added to assure the satisfaction of the constraint when e_{rpqb} is zero.

$$vt_{rp}^{pick} + T_m v^{place} + T_m V_{max}^{place}\left(\sum_b e_{rpqb} - 1\right) \leq vt_{rq}^{pick} \qquad p \in \mathcal{P}, r \in \mathcal{R}, \qquad \textbf{(5.29)}$$

$$vt_{rp}^{pick} + \frac{T_M}{2} v^{place} \leq Q_{rp} - \sum_{\substack{q \in \mathcal{P}_r' \\ q>p}} \sum_{b \in \mathcal{B}_r} (Q_{rp} - \overline{L}_{rb}^{place}) e_{rpqb} \qquad p \in \mathcal{P}, r \in \mathcal{R}, \qquad \textbf{(5.30)}$$

$$\sum_{\substack{s \in \mathcal{P}_r \\ s<p}} \sum_{b \in \mathcal{B}_r} (\underline{L}_{rb}^{place} + R_{rp}) e_{rspb} - R_{rp} + \frac{T_m}{2} v^{place} \leq vt_{rp}^{pick} \qquad p \in \mathcal{P}, r \in \mathcal{R}, \qquad \textbf{(5.31)}$$

5.5.5 Mission in progress constraints (MIP)

Similar to the previous formulation, MIP constraints are added to reflect the subsequent two situations. First, when a robot r_0 is holding a product p_0, and second, when the robot r_0 is on its way to belt A.

When a robots picks a product, the robot becomes busy transferring the product to belt B. To indicate the end of the transferring action, a time τ is set such that τ equals to $\frac{T_m}{2}$. The time τ then starts to decrease and hence the new mission time is re-defined as:

$$\frac{T'_m}{2} = \max(\tau, 0) \qquad \textbf{(5.32)}$$

The time window constraint should be removed since the product p_0 is already picked by robot r_0. The network also needs to start from p_0, thus the following constraint is added:

$$e_{r_0 0 p_0 0} = 1 \qquad \textbf{(5.33)}$$

Moreover, two other constraints need be modified for p_0 and r_0, these constraints are (5.29) and (5.30).

$$\frac{T_m + T'_M}{2} v^{place} + \frac{T_m + T'_M}{2} V_{max}^{place}\left(\sum_b e_{r_0 p_0 qb} - 1\right) \leq vt_{rq}^{pick} \qquad q \in \mathcal{P}_{r0}, q > p, \qquad \textbf{(5.34)}$$

$$\frac{T'_M}{2} v^{place} \leq \sum_{\substack{q \in \mathcal{P}'_{r_0} \\ q>p}} \sum_{b \in \mathcal{B}_{r_0}} \overline{L}_{r_0 b}^{place} e_{r_0 p_0 qb} \qquad \textbf{(5.35)}$$

Constraint (5.29) becomes (5.34) due to the fact that a product has already been picked which means that vt_{rp}^{pick} is now equals to zero. Hence, the next picking action (vt_{rq}^{pick}) should be after the mission in progress is finished. This is reflected by modifying the current mission in progress according to (5.32). Similar argument applies to changing constraint (5.30) to (5.35).

The second case of MIP is when a robot is in B2A state. In this situation, the next picking time must be greater than a time τ. This simulates the time required by the robot to go to belt A after placing a product.

$$R'_{r0p}\left(e_{r_0 0 p 0} - 1\right) + \frac{T''_m}{2} v^{place} \leq vt_{r0p}^{pick} \qquad p \in \mathcal{P}_{r_0}, \qquad \textbf{(5.36)}$$

Where,

$$T''_m = 2 * max(\tau, 0) \qquad \textbf{(5.37)}$$

$$R'_{r0p} = max\left\{\frac{T''_m}{2} V_{max}^{place} - \underline{L}_{r_0 p}^{pick} \frac{V_{min}^{place}}{V_{max}^{pick}}, 0\right\} \qquad p \in \mathcal{P}_{r_0}, \qquad \textbf{(5.38)}$$

5.5.6 Upper and lower bounds

The conveyer belts have to be constrained to be within a range of acceptable velocities. Acceptable velocities means that a belt cannot run faster than a maximum speed limited by the maximum allowable speed of the motorized mechanism that drives the belt. Moreover, the minimum speed of the pickup conveyor belt is limited by the physical dimension of products, this constaint is expressed in (5.40). The place conveyer belt can assume a velocity of 0 [m/min] because the boxes are generated at a constant displacement, this indeed matches the simplified model assumptions in chapter 3. Constraint (5.41) assures that the packing position is positive because the belt is mono-directional. Constraints (5.42) and (5.43) are the binary constraints on e_{rpqb} and e_{r0q0}.

$$V_{min}^{place} \leq v^{place} \leq V_{max}^{place}, \qquad \textbf{(5.39)}$$

$$\frac{1}{V_{max}^{pick}} v^{place} \leq \omega \leq \frac{1}{V_{min}^{pick}} v^{place}, \qquad \textbf{(5.40)}$$

$$vt_{rp}^{pick} \geq 0, \qquad p \in \mathcal{P}_r, r \in \mathcal{R}, \qquad \textbf{(5.41)}$$

$$e_{rpqb} \in \{0,1\}, \qquad r \in \mathcal{R}, p \in P_r, q \in P'_r, b \in \mathcal{B}_r \qquad \textbf{(5.42)}$$

$$e_{r0q0} \in \{0,1\}, \qquad p \in \mathcal{P}_r, r \in \mathcal{R}, \qquad \textbf{(5.43)}$$

5.6 Simulations and results

Model 1 is considered once more to test different aspects of the optimal solution. Similar consideration were taken to reduce the computational cost. The interesting part of this formulation is adding a positive fixed cost and the objective function is minimized by selecting an adequate number of products to pack into boxes. Figure 5-2 plots the time needed for the solver to find a solution versus the time at which the solver is called. The total time is approximated by fifty-seven minutes. This solution schedules products from the moment they pass by the vision system and before entering the work areas of robots, herein referred to as Solution 1. To obtain a faster solution (Solution 2), the solution adopted for future mission approach can be considered. Specifically, only products such that: $\{p|x_p > L_{1,min}^{pick}\}$ are considered in the optimization problem, the computational cost is plotted in Figure 5-3. When the number of considered products is reduced, the time needed to find the optimal solution is equal to seventeen minutes.

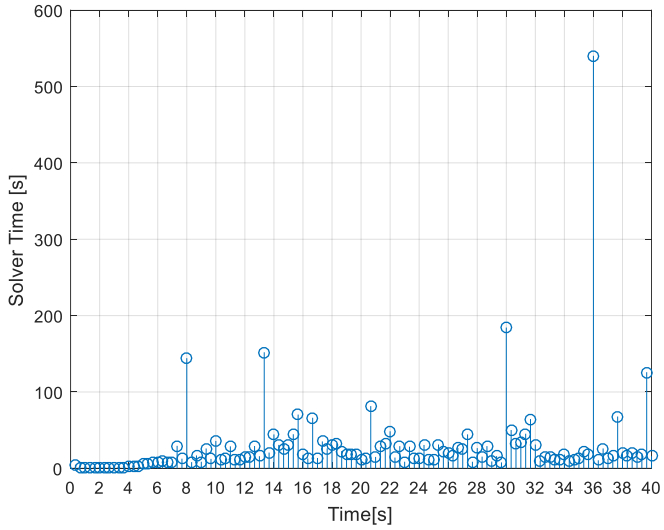

Figure 5-2: Solver cost (Solution 1)

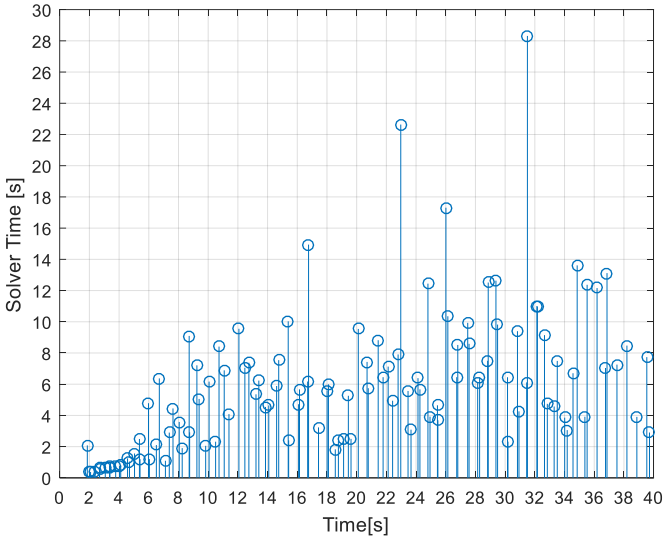

Figure 5-3: Solver cost (Solution 2)

The box conveyor velocity profile for solutions 1 and 2 has similar behaviour in terms of initially "catching up" with products then slowing down to balance the plant. However in solution 1, the velocity reaches a bound of $\left(13.5 \pm 3.5 \left[\frac{m}{min}\right]\right)$. The velocity profile in Solution 2 reaches a velocity of $3 \left[\frac{m}{min}\right]$. This is due to the fact that, solution 2, is trying to compromise between catching up with products and balancing the plant while handling products. Solution 1 instead, makes initial corrections of the solution long before the products and boxes reach the work areas of the robots. Solution 1 and 2 are plotted in Figure 5-4 and Figure 5-5, respectively.

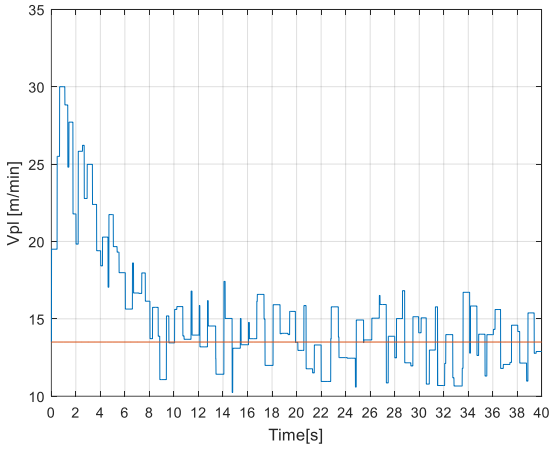

Figure 5-4: Place conveyor belt velocity profile (Solution 1)

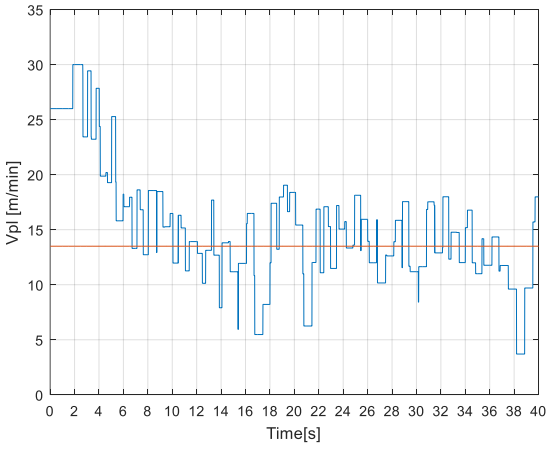

Figure 5-5: Place conveyor belt velocity profile (Solution 2)

Solution 1 and 2 as a consequence have different assignment vectors. Solution 1 tries to assign more products to the upstream robots. Whereas, Solution 2 assigns more products to downstream robots. Table 6 shows a one-hundred-product distribution among robots. The results on the table suggests that for solution 1 robot number 3 is idle most of the time. While

for solution 2, robot 1 is idle most of the time. The time between two consecutive pickups is plotted for both solutions in Figure 5-6 and Figure 5-7.

Table 6: Product distribution

Robot ID	Solution 1	Solution 2
1	40	29
2	37	37
3	23	34

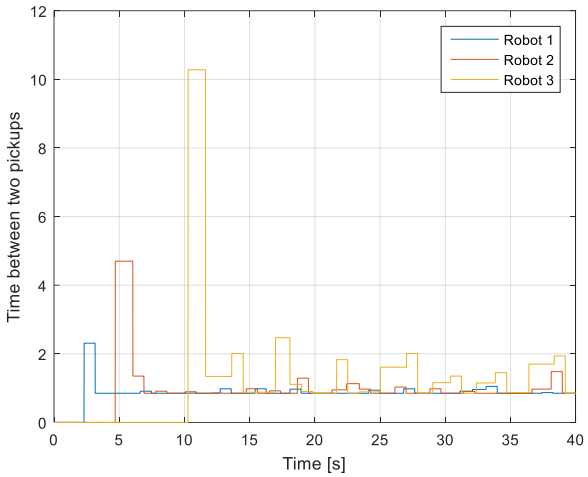

Figure 5-6: Time between two pickups (Solution 1)

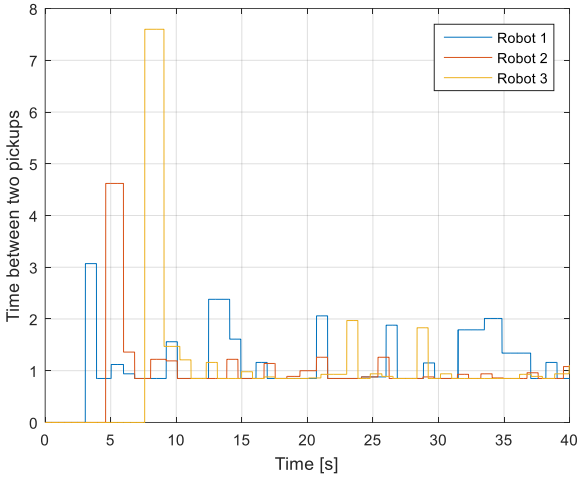

Figure 5-7: Time between two pickups (Solution 2)

These observations are reflected on the assignment of products to robots as in Figure 5-8 and Figure 5-9.

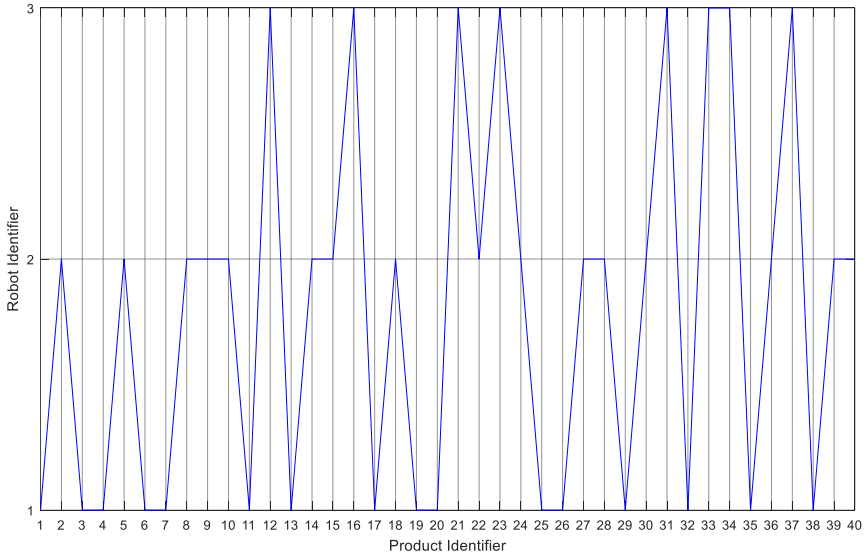

Figure 5-8: Products' assignment (Solution 1)

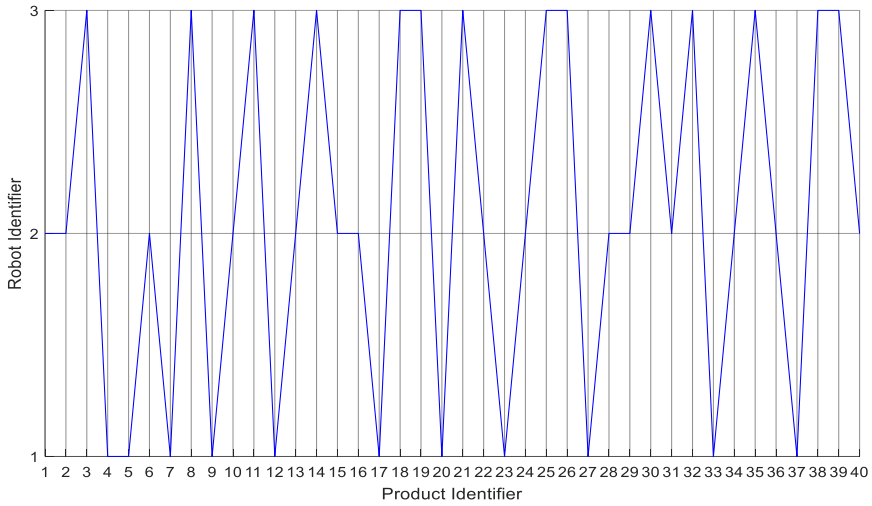

Figure 5-9: Products' assignment (Solution 2)

The number of lost products in the start-up transient is successfully reduced which shows a superiority of this modelling approach over the cyclic assignment. The two proposed solutions reduced the number of lost products.

Table 7: Comparing scheduler performance in terms of the number of lost products

Control technique	**Lost prod.**
Optimal Line Manager – Network Flow (Solution 1)	0
Optimal Line Manager – Network Flow (Solution 2)	0
Cyclic assignment	9

5.7 Conclusion

A second MILP formulation is constructed as a network flow problem. This solutions is implemented on model 1 and it showed its superiority over the cyclic assignment. The manager increased the speed of the place conveyor and eventually the belt's velocity reached the expected value such that the plant is self-balanced. At this status of the controller, the manager cannot be implemented on a real-time system. This is because it does not abide by the industrial standard of finding a solution in less than a second. The solution, however, can still be implemented as an offline solution. Next chapter presents a heuristic solution that takes advantage of the results obtained from the MILP formulation.

6 Simulation based optimization

6.1 Introduction

The motivation of this solution is based on a number of important results that were observed from studying the optimal problem. Due to the fact that the boxes can hold different number of products, the flow of boxes and products is initially imbalanced which leads to losing products. Since boxes can hold several products, it is often the case where the flow of boxes is slower than the one of products to reach a self-balancing plant. Due to the initial imbalance of the plant, the robots cell with a cyclic assignment strategy fails to manipulate all entering products. To overcome this problem, belt B is filled with a number of empty boxes then products are allowed to flow into the system. This way the initial imbalance is reduced and more products are packed. To optimize the plant performances with this heuristic approach, the answers to the following questions have to be investigated and studied in a systematic way: *"how many boxes should initially be present on the conveyor*?" and *"which kind of assignment of products and boxes to robots has to be considered?"*

As it is evident, it is necessary to consider an automatic procedure to test all possible combinations of initial boxes and assignments. This is possible due to the availability of the simplified model that has been previously discussed. In order to avoid testing an infinite number of possibilities, a systematic procedure is required.

A natural solution is to adopt the cyclic assignment as a good candidate for a steady state solution. Then, several number of modifications are carried out in order to improve the assignment strategy at the start-up transient.

To improve the plant performances during the start-up transient with respect to a generic system parameter x (i.e. the box distance or the number of empty boxes initially present on the place conveyor), a Simulation Based Optimization (SbO) has been studied. Specifically the plant performances are evaluated considering the number of lost products during the start-up transient. This performance index can be considered as the cost function to be minimized. This function has to be dependent on the "free" parameter x (Changing the value of x, the number of lost products has to be different). For the selected parameter x, suitable bounds have to be specified, according to physical constraints. Specifically the search for the optimal value of a plant cost function has to be carried exploring the interval $[x^{min}, x^{max}]$. To find the best solution, the interval is divided into a fixed number of values. For every possibility, a start-up

simulation is performed. The best $\overline{x}$ solution is the one which leads to the lower number of lost products. The granularity (discretization of the interval, Δx) has to be small enough to select solutions close enough to the optimal one. However, a greater number of explored solutions leads to a greater computational effort. Considering sufficiently small Δx, a value x^* is expected to be reached. This value is not, necessarily, equal to the optimal solution $\overline{x}$, but it should be close enough. A schematic representation of this heuristic procedure to optimize the plant performances is presented in Figure 6-1.

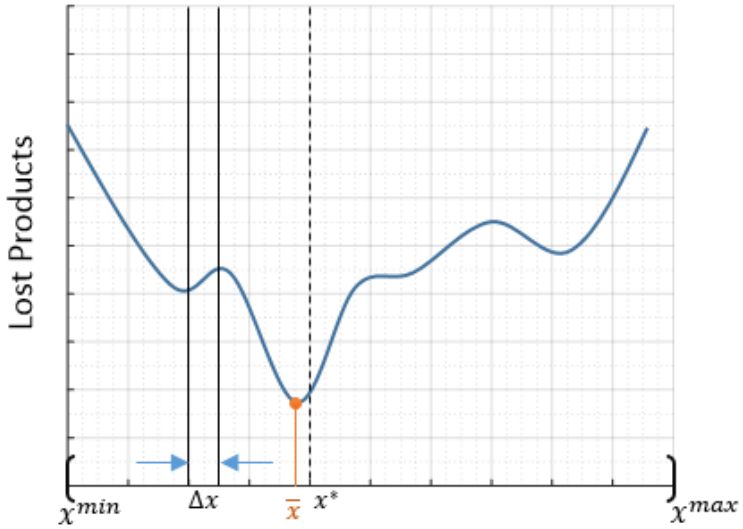

Figure 6-1: Finding the optimal solution of a "free" parameter of a generic plant

This chapter discusses, first, the situation at which the cyclic assignment is sufficient to balance the plant and the limitation to this approach. Second, the effect of initializing the robots cell with empty boxes is justified. Third, starting with an arbitrary schedule then switching to a cyclic one is presented. Finally, a suitable combinations of these modifications to reduce the number of lost products is presented.

6.2 Cyclic assignment solution

This solution is advantageous in terms of a periodic steady state solution. However, this solution performs poorly at the initial state and the solution is worsen when the last downstream robot is placed far away from the first upstream one. To demonstrate this observation, given that Δx_b is the distance between two boxes, the velocity of boxes' conveyor is computed such that the system is self-balanced as the following

$$V_{pl} = \frac{N_{row}^{p}}{N_{layer}^{b}} \Delta x_b \; PR \qquad \textbf{(6.1)}$$

The distance between two products $\Delta x_p = N_{row}^p \frac{V_{pi}}{PR}$, and the distance of the first product after the first box reaches L_{place} is given by:

$$x_p^1 = L_{place} \frac{V_{pi}}{V_{pl}} \quad \textbf{(6.2)}$$

Hence, the number of leading products (N_{Lead}^p) is given by:

$$N_{Lead}^p = \left\lceil \frac{x_p^1 - L_{pick}}{\Delta x_p} \right\rceil = \left\lceil \frac{PR}{N_{row}^p V_{pi}} \left(x_p^1 - L_{pick}\right) \right\rceil = \left\lceil \frac{PR}{N_{row}^p} \left(L_{place} \frac{1}{V_{pl}} - L_{pick} \frac{1}{V_{pi}} \right) \right\rceil \quad \textbf{(6.3)}$$

In order to reduce N_{Lead}^p to zero, the velocities are set such that:

$$\frac{V_{pi}}{V_{pl}} = \frac{L_{pick}}{L_{place}} \quad \textbf{(6.4)}$$

Combining equations (6.1) and (6.4) leads to:

$$\Delta x_b = \frac{L_{place}}{L_{pick}} \frac{N_{layer}^b}{N_{row}^p} \frac{V_{pi}}{PR} \quad \textbf{(6.5)}$$

$$\Rightarrow \Delta x_b = \frac{N_{layer}^b}{N_{row}^p} \frac{V_{pi}}{PR}, \qquad Given\ L_{place} = L_{pick} \quad \textbf{(6.6)}$$

Equation (6.6) assures a complete balance of the flow of boxes and products. The following figures demonstrate the idea by considering three cases. In the first case, the number of rows of products is equal to the number of layers which suggest that the spacing of the boxes should be the same to that of products, Figure 6-2. In the second case, the number of rows are larger than the number of layers which leads to the spacing of boxes to be half of that of the products, Figure 6-3. Case three is the opposite scenario where the spacing of boxes are the double of that of the products, Figure 6-4.

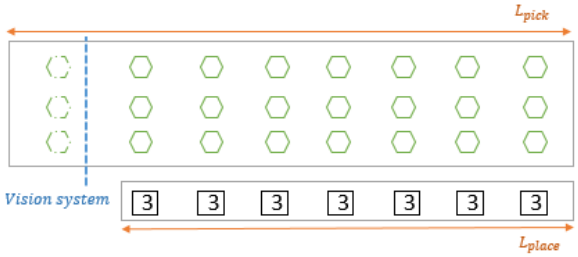

Figure 6-2: Equal number of rows and layers (Case 1)

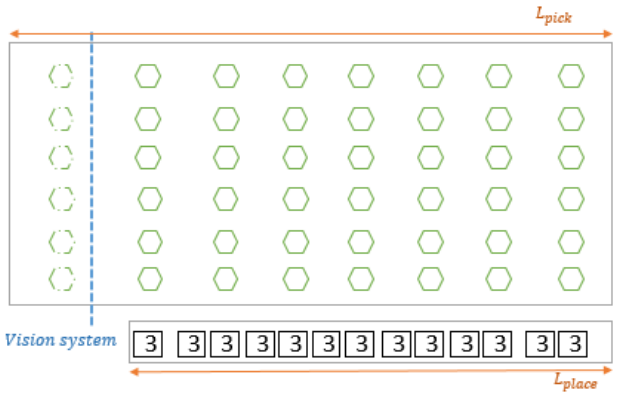

Figure 6-3: Number of rows is more than the number of layers (Case 2)

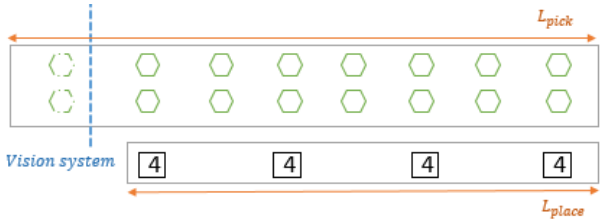

Figure 6-4: The number of layers is more than the number of rows (Case 3)

The proposed solution assures a complete product manipulation when the products are assigned cyclically to robots. However, there is a limitation to this approach. In the case where the ratio $\frac{N_{layer}^{b}}{N_{row}^{p}}$ is greater than 1 and if the number of robots is less than N_{layer}^{b}, then a cyclic strategy fails to manipulate the number of passing products given that the conveyer velocities are too fast. This is because each robot has a particular window of time within which a product can be handled. Henceforth, increasing V_{pi} shortens this window and few products are picked. A similar scenario where the ratio $\frac{N_{layer}^{b}}{N_{row}^{p}}$ is less than 1 and N_{row}^{p} is larger than the number of robots. As a solution, the time window of a robot could be increased by slowing down the conveyor velocities. However, the minimum velocities are limited by the physical dimension of boxes and products.

Evidently, the cyclic assignment alone is not sufficient to obtain a general solution for different plant layouts. Therefore, a number of improvements to the approach is required. The next sections describe extensions to this algorithm. Hereinafter, the term optimization is used loosely to refer to the procedure of finding the best solution among a closed set of possible solutions.

6.3 Optimization over IBD

As said before, Δx_b is not usually a completely free parameter and the solution proposed in section 6.2 (based on the reduction of leading products) may not be sufficient to reach suitable plant performances during the start-up transient. In those situations, a simulation based optimization can be performed considering the Initial Box Distance (IBD) as a "free" parameter to be selected. As mentioned before, this solution consists in placing some empty boxes on the place conveyor before the products start entering the packaging line. In Figure 6-5, the selected IBD leads to the presence of three empty boxes on the place conveyor.

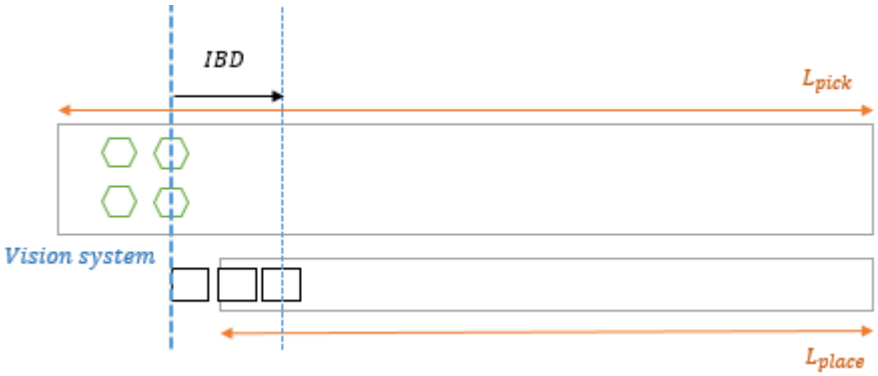

Figure 6-5: Initial Box Distance (IBD)

The IBD is the position of the first box on the place conveyor when the products start entering the packaging line. Formally, consider once again (6.3) and subtract IBD from L_{place} as the following:

$$N_{Lead}^{p} = \left\lceil \frac{PR}{N_{row}^{p}} \left((L_{place} - IBD)\frac{1}{V_{pl}} - L_{pick}\frac{1}{V_{pi}} \right) \right\rceil \qquad \textbf{(6.7)}$$

In this case, Δx_b is fixed as well as V_{pl}. On the contrary, IBD in (6.7) is a free parameter. In this case, letting N_{Lead}^{p} equals to zero does not necessarily assure zero loss of products due to the fact that V_{pl} is either slower or faster than V_{pi}. Nonetheless, reducing N_{Lead}^{p} essentially reduces the number of lost products (N_{lost}^{p}). Under no condition, a box is allowed to leave the belt without being completely full. This is satisfied by stopping the belt whenever a partially filled box reaches a distance equals to L_{place}. This essentially adds an undetermined V_{pl} profile that is hard to predict. The following question naturally arises: *"What is the value of IBD such that N_{lost}^{p} is minimized?"*. To answer this question, a cyclic scheduling is considered and the optimal value of the IBD is selected exploring a significant number of possibilities and choosing the one with the lower number of lost products. This method is referred to as optimization over IBD. This optimization procedure starts from considering a cyclic

assignment for both boxes and products. Then, the plant behaviour during the start-up transient is simulated. The end of the start-up transient is computed considering the definition introduced in section 3.4.3. Specifically, the duration of the simulation has to be greater than the time instant τ_s when the number of lost products reaches a constant value. This time decides the maximum simulation period that one should consider for the proceeding tests. Once the experiment time is decided, the experiment is carried out with an incremented value of IBD. The procedure is repeated for a selected values of IBD. The upper bound of IBD is the length of the place conveyor belt. This algorithm is summarized according to the following:

1- Incoming products and boxes are assigned cyclically and the IBD is set to zero
2- A simulation of the start-up transient is conducted and the total number of lost products, given that the plant is self-balanced at steady state, is recorded
3- The time τ_s at which the number of lost products stopped increasing is recorded
4- The start-up transient simulation is repeated for a simulation period larger than τ_s for all the values of IBD in a range of [0,IBD_{max}], with a discretization interval of ΔIBD
5- After each experiment, the number of lost product is recorded

The optimal value of IBD (IBD^*) is selected as the one which minimizes the number of lost products.

6.4 Optimization over the Switching Instance (SI)

Optimization over IBD alone might not be enough for the general case. For those cases, Changing the schedule as well as IBD might optimize the plant. The idea is to start with an arbitrary schedule, then after a while switch to a cyclic one. To generate such an arbitrary schedule, the robots are considered to be greedy which means that each robot picks any existing product in its work area given that the products are not assigned to any other robots. This greedy rule is formalized according to the following:

$$S_r = \{p|\ L_{r,min}^{pick} \le x_p \le L_{r,max}^{pick}\ AND\ (p \notin S_{r'}, \forall r' \neq r)\ \}, r \in \mathcal{R}, p = 1,2,\dots.P \qquad \textbf{(6.8)}$$

Where,

S_r: is the set of products that are assigned to robot r

Initially, products are not assigned to any robots which means that the assignment sets for each robot (S_r) is empty. As the products are entering the work areas of robots, this set is filled with all products within the robot's work area. Each robot can handle up to a maximum number of products at a given window of time. Then, some products are left to the next robots of the packaging line. This method generates an arbitrary assignment of products with no guarantee

of a steady state behaviour. Therefore, after applying this rule for the first n products, a cyclic assignment is applied. In the scenario of Figure 6-6, the first six products are not assigned to any robots. Hence, the first robot would handle the maximum possible number of passing products, for instance two. As a result, four products would leave the work area of the first robot. The second and the third robots would do the same. For all the products after the sixth one, each robot would respect the cyclic assignment. This optimization algorithm tries to find the proper number of products n such that the number of lost products is minimized. This method is effective when there exist a number of empty boxes to host the picked products. Hence, combining this algorithm with IBD is justified. Herein, this method is referred to as optimization over the Switching Instance (SI).

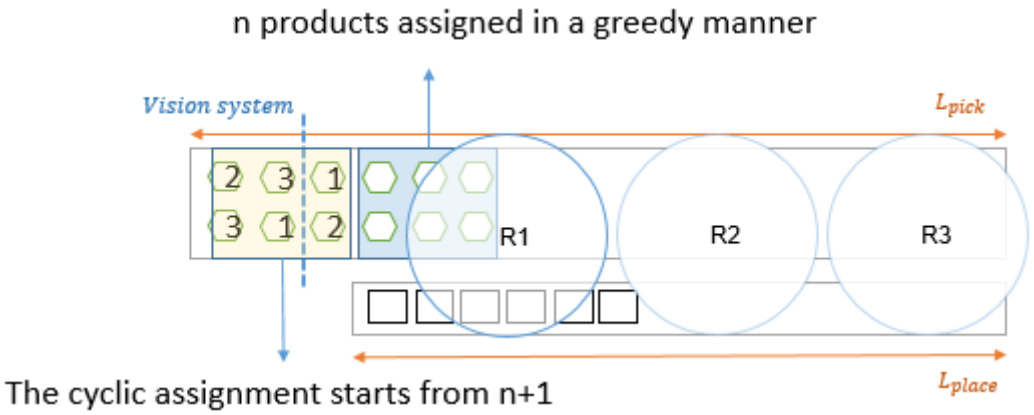

Figure 6-6: Assigning the first n products in a greedy manner

6.5 Combining SI and IBD solutions

Assuming that the plant is sized such that the nominal mission time of robots is sufficient to satisfy the maximum product rate. That is:

$$T_m^{nom} = \frac{PR}{60*R}[pps/robot] \quad \textbf{(6.9)}$$

Each robot, however, needs to be able to perform the task in a time period less than the nominal one. This extra requirement avoids saturating the robots, and thus the robots can handle extra products when needed. Therefore, the minimum mission time for a robot can be, for instance, twenty percent less than the nominal one, as expressed by equation (6.10).

$$T_m = 0.8\, T_m^{nom} \quad \textbf{(6.10)}$$

Then, the following algorithm is applied to minimize the number of lost products:

1- Incoming products and boxes are assigned cyclically and the IBD is set to zero

2- A simulation of the startup transient is conducted and the total number of lost products at steady state is recorded
3- If the steady state condition is not reached after $T_{simulation}^{MAX}$, the algorithm stops
4- The time τ_s at which the number of lost products stopped increasing is recorded
5- The startup transient simulation is repeated for a simulation period larger than τ_s for all the values of IBD in a range of [0,IBD_{max}], with a discretization interval of ΔIBD
6- After each experiment, the number of lost product is recorded
7- The optimal value of IBD^* is selected as the one which minimizes the number of lost products
8- If N_{loss}^p = 0, then stop. Else 9
9- The startup transient simulation is repeated for all the values of n from 0 to n^{MAX}, where n is the number of products assigned to robots with greedy strategy (after this n "greedy products" switch to cyclic assignment).
10- If N_{loss}^p = 0, then stop, else 11
11- Increase Δx_b and compute IBD^{NEW} with the method presented in section 6.5.1 (re-computation of IBD)
12- If maximum number of iteration $ITER^{MAX}$ is reached or the number of lost products N_{loss}^p is equal to zero, Then Stop, Else go to 9

The algorithm is summarized in the flow chart in Figure 6-7.

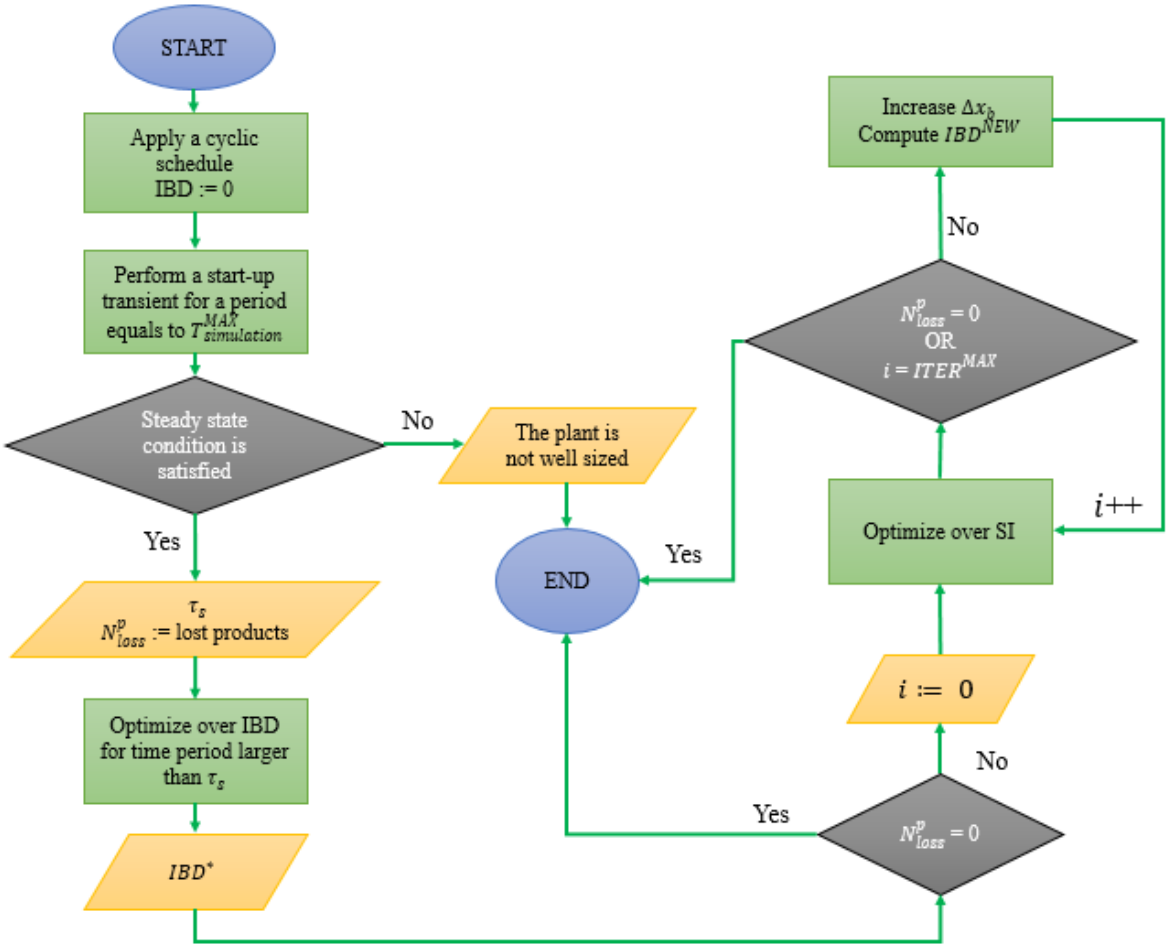

Figure 6-7: Algorithm to optimize generic plant layouts

The advantage of this algorithm is to find the minimum number of lost products in an iterative manner instead of trying all possible combinations of IBD, SI and Δx_b. This algorithm optimizes the plant with respect with one parameter at a time and then proceeds to the other. The sequence of operations should be respected due to that fact that optimizing over SI depends on having boxes initially present on the belt. Hence, there is no point of starting with this variable. For the same reason, Δx_b should be left to the end since boxes should be available for the effectiveness of the optimization over SI. Consider the following plot of the number of lost products as a function of SI and IBD. According to Figure 6-8, the function shows a valley along IBD. The figure also shows that for a certain value of IBD the solution along the SI direction experiences peaks and valleys, this is another evidence that the solution should start with IBD. This discussion is completed in the simulation and results section.

-

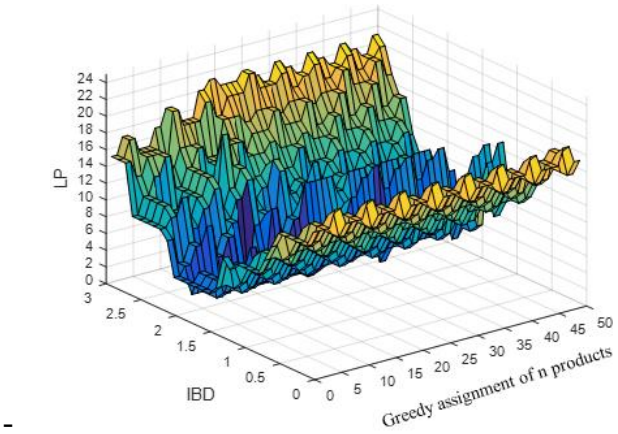

Figure 6-8: Lost products versus (IBD,SI)

6.5.1 Estimation of IBD

As a result of updating Δx_b the place conveyor velocity changes proportionally. At this point, the previously obtained IBD is no longer the position at which the number of products is minimized. For the optimization procedure to continue, a new value of IBD needs to be obtained. Instead of running another set of experiments for obtaining a new IBD, computing a new value of IBD as a function of the old one is advantageous. It is realized that when increasing Δx_b, V_{pl} increases in order not to lose the self-balancing condition. This means that the new IBD should be less. To obtain this new IBD, a point on the conveyor is fixed at which the first product meets the first box. This point is referred to as the meeting point X_{mp} as shown in Figure 6-9.

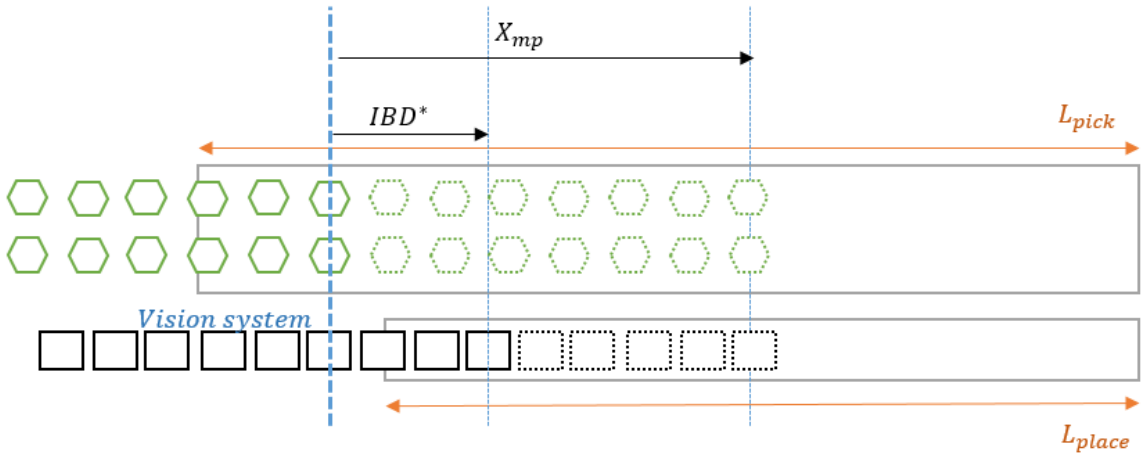

Figure 6-9: IBD is depicted along with the meeting point X_{mp}*

By keeping this point constant for each update of Δx_b, a new estimate of IBD is obtained. Given the pick and place conveyor belt velocities, the meeting point is given by:

$$X_{mp} = \tau\, V_{pi} \qquad \textbf{(6.11)}$$

$$X_{mp} - IBD^* = \tau\, V_{pl}^* \quad \textbf{(6.12)}$$

Where,

τ: is the time after which the first product meets the first box at X_{mp}

IBD^*: The currently obtained IBD

V_{pl}^*: the current conveyor belt velocity obtained as a function of the current Δx_b

Rearranging (6.11) and (6.12) leads to:

$$X_{mp} = \frac{V_{pi}}{V_{pi} - V_{pl}^*} IBD^* \quad \textbf{(6.13)}$$

The new value of IBD as a function of X_{mp} is given by:

$$IBD^{new} = \frac{V_{pi} - V_{pl}^{new}}{V_{pi}}\, X_{mp} \quad \textbf{(6.14)}$$

Hence the new IBD as a function of the old one is given by:

$$IBD^{new} = \frac{V_{pi} - V_{pl}^{new}}{V_{pi} - V_{pl}^*}\, IBD^* \quad \textbf{(6.15)}$$

Where,

V_{pl}^{new} : is the new place conveyor belt velocity obtained as a function of the new updated Δx_b

This method of computing the new value of IBD avoids the repetition of the optimization over IBD step. Consequently, the optimal solution is obtained in fewer steps.

6.6 Simulations and results

6.6.1 Numerical parameters

In this case, the plant under test is Model 3 where six robots are manipulating two rows of products and four layers per box. The model parameters are listed in the following table:

Table 8: numerical values for the parameters of MODEL 3

Parameter	**Value**
Vision system Position	20 [cm]
PR	120 [ppm]
$L_{1,min}^{pick}$	120 [cm]
$L_{6,max}^{pick}$	764 [cm]
$L_{1,min}^{place}$	130 [cm]
$L_{6,max}^{place}$	772 [cm]

$L_{r,max}^{pick} - L_{r,min}^{pick}$	70 [cm]
$L_{r,max}^{place} - L_{r,min}^{place}$	60 [cm]
$L_{r-1,max}^{pick} - L_{r,min}^{pick}$	20[cm]
$L_{r-1,max}^{place} - L_{r,min}^{place}$	20[cm]
$\Delta x_{b,min}$	45 [cm]
V_{pi}	30 [m/min]
V_{pl}	13.5 [m/min]
T_m	1.46 [s]

6.6.2 Effect of the distance between two boxes (Δx_b)

The first result is regarding the effect of changing Δx_b on the number of lost products when a cyclic schedule is assumed. Figure 6-10 shows that there is a tendency of reduced N_{loss}^{p} when Δx_b is increased. Each vertical line of the figure represents the total number of lost products after a start-up simulation period of two-hundred seconds. Increasing Δx_b increases V_{pl} which reduces the difference between the flow of the products and boxes. According to equation (6.6), Δx_b should be equal to $50[cm]$ and V_{pl} equals to V_{pi}. However, the robot mission time is high which requires that V_{pi} to be low. Due to the dimension of the products entering this plant layout, V_{pi} cannot be reduced any further. Hence, the solution of section 6.2 cannot be applied. Instead V_{pl} is computed to assure a self-balancing plant. Figure 6-10 shows that manipulating Δx_b with a cyclic assignment is not enough to obtain a zero loss of products. The figure also shows a non-uniform behaviour for Δx_b greater than 60[cm] which means that there is no need to consider large values of Δx_b beyond 60 [cm].

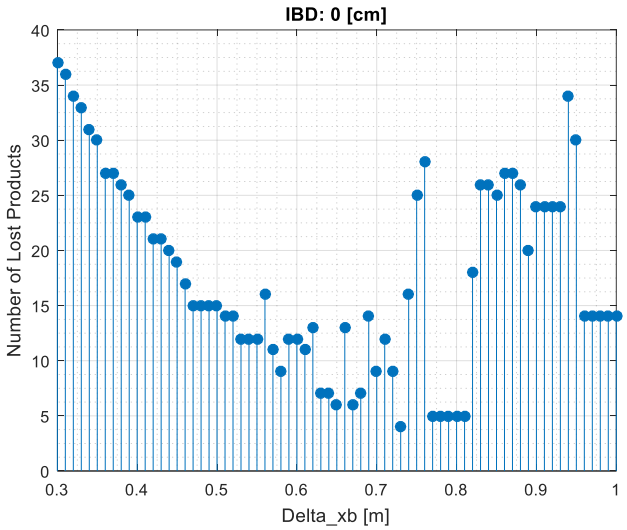

Figure 6-10: effect of Δx_b on the number of lost products

6.6.3 Optimizing over IBD and SI

Previously argued that the algorithm in Figure 6-7 is the shortest path to find the minimum number of lost products. Although a formal prove is difficult to formalize, some simulation results support the assumption for the particular case of model 3. Without following the proposed algorithm, one would try a large set of possible combinations of IBD and SI. Figure 6-11 plots the effect of changing IBD and SI on the number of lost products. Dark areas on the contour plot corresponds to less number of lost products. To obtain this plot the experiment is conducted as the following:

1- A cyclic scheduling is assumed
2- The total number of lost products are recorded for a simulation period of 400 [s]
3- The experiment is repeated for an IBD range of [0,3] with a step of 0.1 [m],
4- If n >50, END
5- The cyclic assignment is assumed after the n^{th} product, the range of products up to the n^{th} product is manipulated by the first greedy robots
6- Increment n by 1,Go to 3

The contour in Figure 6-11 shows a small range of IBD such the number of lost products is minimized. It is also shown that if no arbitrary scheduling is applied, N_{loss}^{p} cannot be equal to zero. The plot shows that starting with an optimization over an IBD is superior to the optimization over SI because moving on the direction of IBD leads to a clear set of local minima.

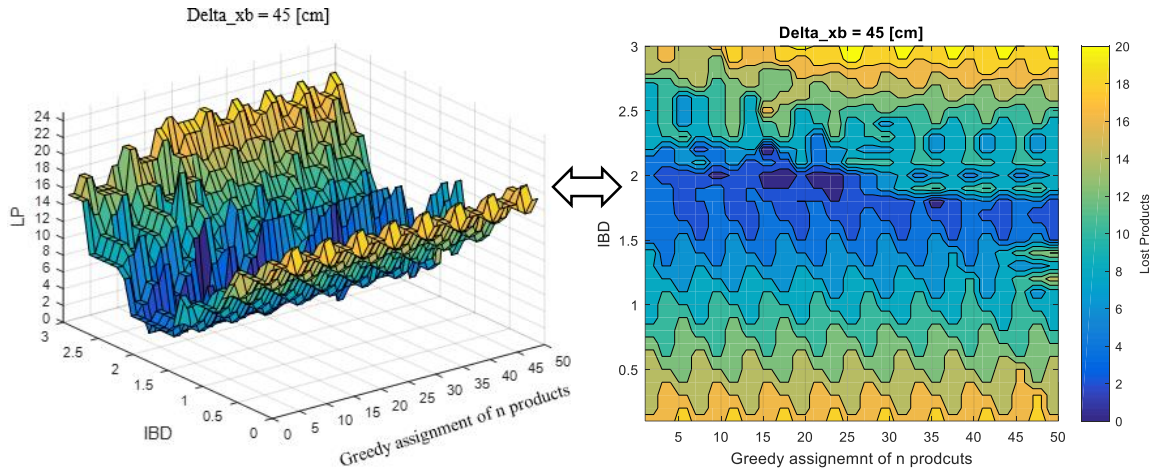

Figure 6-11: Optimization over IBD and SI with Δx_b= 45 [cm]

It is noted, however, the proposed algorithm might reach a sub-optimal point as shown in Figure 6-12. This point corresponds to an IBD of 1.6 [m] and an SI after 7 products. When the sub-optimal solution happens the algorithm extend the distance between boxes and computes a new estimate of IBD according to (6.15). In this case Δx_b is increased by 10 [cm] and V_{pl} becomes 16.5 [m/min]. With these new inputs, the estimate of the corresponding IBD is equal to 1.2 [m]. Figure 6-13 depicts the number of lost products as a function of IBD and SI. It is noted that the range of values of IBD that minimizes the number of lost products is correctly estimated for the new value of Δx_b (see Table 9). The algorithm as a result finds the solution that corresponds to the white cross, which is the optimal point of zero lost products.

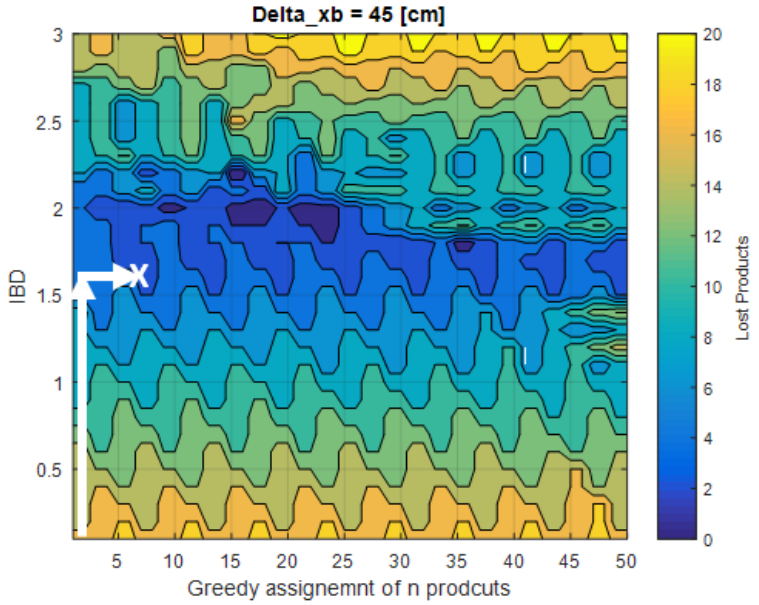

Figure 6-12: The arrows correspond to the solution path found by the algorithm

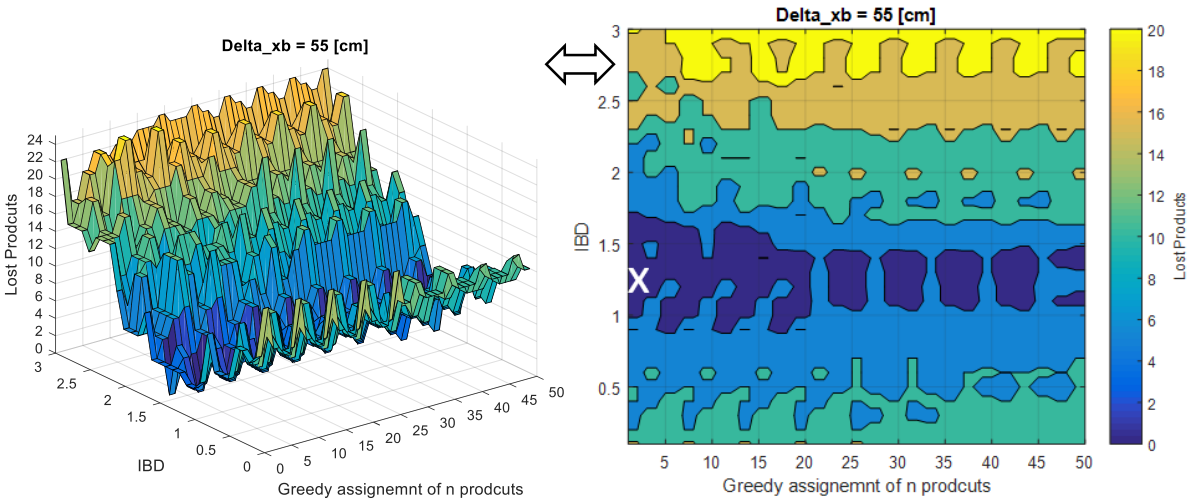

Figure 6-13: Optimization over IBD and SI with Δx_b = 55 [cm]

Table 9: IBD with Δx_b of 45 [cm] and IBD^{new} computed with Δx_b of 55 [cm]*

$\boldsymbol{IBD^*}$ [m]		$\boldsymbol{IBD^{new}}$ [m]	
Min	Max	Min	Max
1.5	2.3	1.13	1.69

Conveniently, for small plant layouts with less number of robots the optimization algorithm might find the algorithm with just optimization over IBD. This case is illustrated on models 1 and 2 as shown in Figure 6-14 and Figure 6-15, respectively.

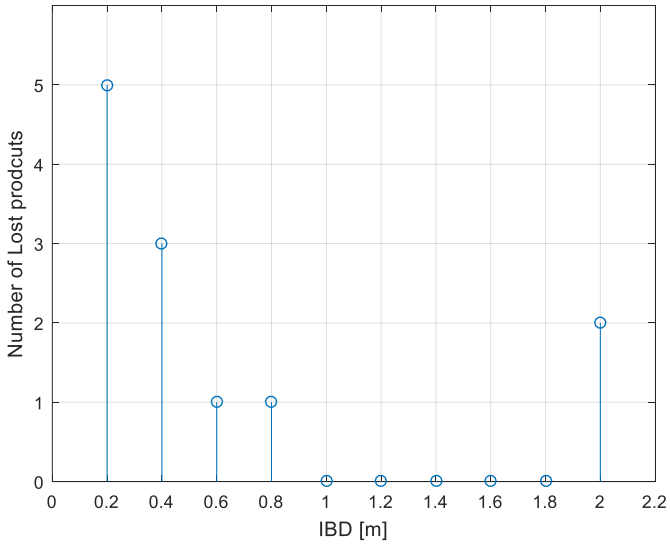

Figure 6-14: Optimization over IBD (Model 1)

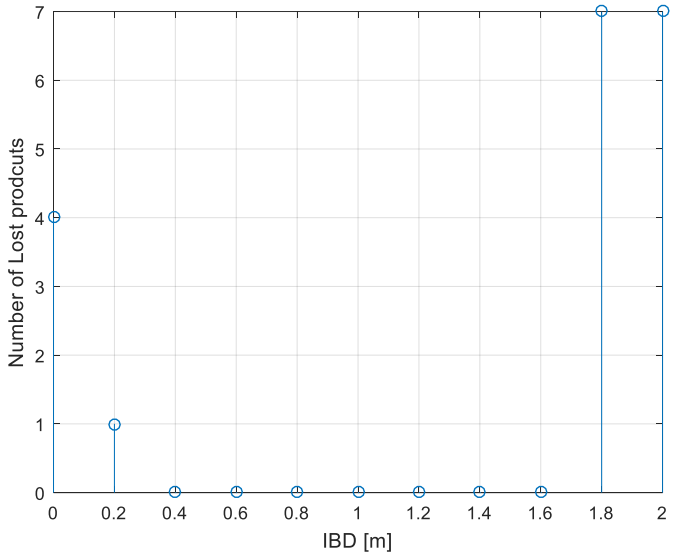

Figure 6-15: Optimization over IBD (Model 2)

6.7 Conclusion

This chapter presented a heuristic approach to find an optimal controller of the plant. The method is based on a widely adopted control technique for co-flux plant layouts, namely the cyclic assignment. An approach to improve the cyclic assignment in terms of start-up transient is proposed. The technique initializes the place conveyor with a number of empty boxes in order to reduce the number of leading products. Another control strategy assigns the first n products in a greedy manner then the following products are assigned cyclically. This strategy is shown to be functioning effectively, in conjunction with an initial number of empty boxes. Furthermore, a thorough algorithm was proposed to optimize the plant with respect to one manipulated variable at a time. The sequence of the algorithm was supported with simulation results.

7 Comparisons and results

7.1 Introduction

Up to this point, different controllers have been considered to control co-flux packaging lines. A basic control algorithm has been presented as a good candidate for a steady state solution, namely the cyclic strategy. During the start-up transient, however, this strategy fails to manipulate all passing products due to the initial imbalance of the system. To obtain better results, the problem was modelled in a closed loop configuration. The problem was posed as a mixed integer liner program optimization problem and applied to the plant as model predict controller with a receding horizon. The motivation of this, is to control a generic plant against unknown disturbances while maintaining a minimum loss of products. Without this closed-loop configuration, it is difficult to manage the scheduling of the products and boxes such that the number of lost products is minimized.

The focus, as a result, is oriented towards solving the start-up transient problem. The advantage of the MILP solutions lies in the fact that the procedure is fully automatic and no special attention to the behaviour of the solution is required. However, the high computational cost, makes the solution infeasible to implement on an industrial solution, as shown in chapters 4 and 5. Therefore, a heuristic solution is developed to be feasibly implementable on a real system. The solution should also be reproducible and expandable to the general case. That is given a generic plant, a systematic procedure needs to be followed in order to obtain the minimum number of lost products during the start-up transient, as shown in chapter 6.

This chapter highlights a general comparison among the proposed solutions in chapters, 4, 5 and 6. The comparison is carried out in terms of the total number of lost products, computational cost and finally the convergence to a steady-state behaviour.

7.2 Comparisons in terms of lost Products in start-up transient

Regarding the number of lost products, the following table, Table 10, lists the performance of the three proposed solutions on different three plant models. For easy referencing, the solution in chapter 4 is referred to as FM-OLM, whereas the solution of chapter 5 is referred to as NF-OLM. The procedure in chapter 6 is referred to as SbO.

Table 10: Comparison in terms of the total number of lost products

Scheduler	Vpi [m/min]	PR[ppm]	R[#]	N^b_{layer}[#]	N^b_{row}[#]	N^p_{loss}[#]	Model
Cyclic	26	180	3	2	2	9	Model 1
FM_OLM						0	
NF-OLM						0	
SbO						0	
Cyclic	40	180	3	2	2	15	Model 1.1
FM_OLM						3	
NF-OLM						2	
SbO						0	
Cyclic	26	180	3	4	2	34	Model 1.2
FM_OLM						2	
NF-OLM						2	
SbO						0	

The table presents three test cases. All of which has the same number of robots. The first test case regards model 1 which is the previously described model. Model 1.1 represents the same model with different velocity of the pick conveyor belt. This way the number of lost product is increased. The third test case regards the change in the number of layers per box which means that the boxes are initially traveling at slower rate than products.

7.3 Comparisons in terms of computational cost

In terms of computational cost, the comparison is relevant to the online solution. Both versions of the MILP formulation suffers from high computational cost. It has been shown that considering less number of products results in a faster solution. However, larger plants assumes larger number of decision variables and leads to an incredibly high computational cost. To put things into perspective, Table 11 tabulates the computational cost for finding the optimal solution of model 1 using the three proposed controller. The table shows that SbO finds the solution in a shorter time.

Table 11: Computational cost for finding the optimal solution on Model 1

Controller	Simulation period [s]	Computational Cost [s]
Network Flow	40	1020
Future Mission	40	275
Simulation based Optimization (SbO)	40	132

7.4 Superiority of SbO over MILP solution

With the simulation based optimization in all test cases, the number of lost products is minimized to zero. This shows the superiority of this technique over the optimal problem. Since SbO starts from a steady state solution, the steady state solution is guaranteed to be reached. This is not true for the optimal problem since there is no constraint that guides the solution to a particular behaviour. Instead the solution keeps correcting its decision and it may over compensate and loose few more products as in models 1.1 and 1.2. The common factor between these two solutions is that in both systems at every occurrence of an event the number of holes are excessively many compared to the number of products, refer to Figure 7-1. Similar argument applies to model 1.2; it is true that there are more products in the system but the number of layers now is doubled, refer to Figure 7-2.

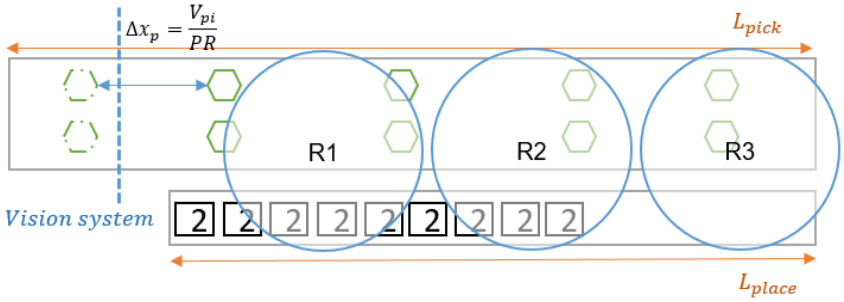

Figure 7-1: Number of holes is more than the number of products in the system, (Model 1.1)

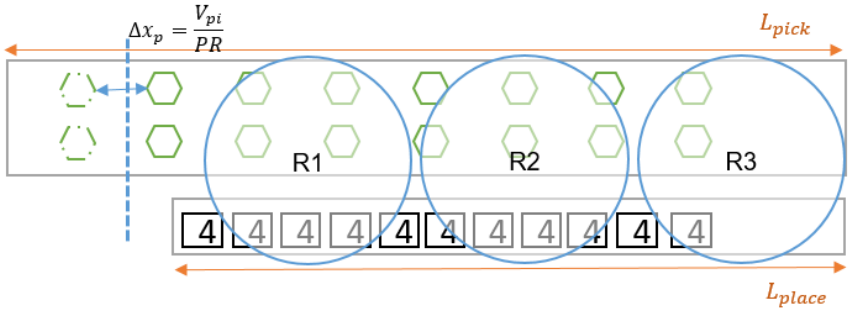

Figure 7-2: Number of holes is more than the number of products in the system, (Model 1.2)

The last comparison is in terms of the steady state behaviour. Consider the time between two consecutive pickups for each robot in model 1. The simulation based optimization results in a control strategy that is able to distribute the load of products equally among the team of robots. Figure 7-3 illustrates this behaviour in terms of the time between two pickups.

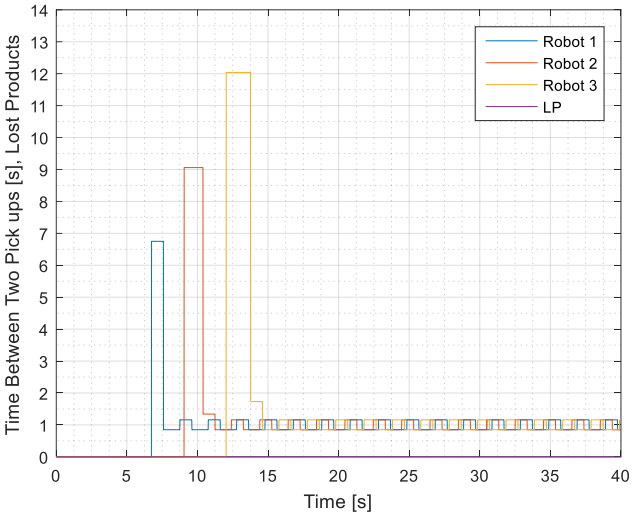

Figure 7-3: Time between two pickups (SbO, Model 1)

Figure 7-4 illustrates the time between pickups as a result of the MILP solution. The behaviour illustrates an imbalance distribution of products among robots.

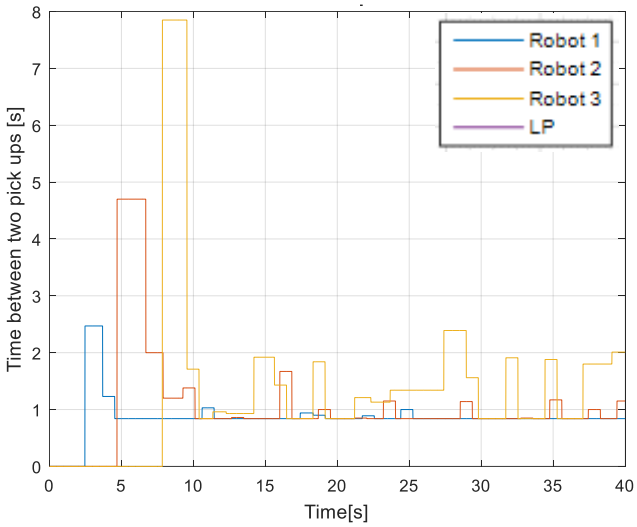

Figure 7-4: Time between two pickups (MILP, Model 1)

7.5 Conclusion

This chapter highlighted a general comparison among the proposed control strategies in chapters 4, 5 and 6. Due to the computational infeasibility of the models in chapters 4 and 5, the comparison was carried out in terms of small-sized plants. The simulation based optimization have shown a superior behaviour in terms of the number of lost products and the steady state behaviour. Therefore, SbO is applied to the extended model of the next chapter where a multi-grip robot model is proposed.

8 Extensions to multiple-grips robot plants

8.1 Introduction

Previously in chapter 3, a simplified model was developed for plant layouts with mono-grip robots. This model is used to approximate the behaviour of plants with either mono or multiple grips. For these types of plants, test results showed the efficacy of this modelling approach for both models. However, for the multi-grip case an extra condition is imposed on the scheduling strategy. This condition regards the scheduling of a group of products to one robot. This assumption is clearly a limitation to the general case of scheduling any combination of products in that group. This chapter presents a further extension to the simplified model of chapter 3. The model is validated against one model plant that matches a real case. Finally, the controller of the plant is developed following the algorithm introduced in chapter 6.

8.2 Robot model

The model is extended by appending an extra state at which the robot has one product on one grip, the state is referred to as (Pone). The robot travels to the next product within a minimum picking time, defined as T_p. Once all grips are filled with products, the robot moves from belt A to B. Similar to the previous model, the robot takes a minimum of one half of a mission time equals to $T_m/2$ to travel between belts. At this stage, the robot is waiting for a box. Once the box enters the placing area, the robot deposit the whole grip into one layer. For the other half of the mission time, the robot travels back to belt A, and to the initial state (W4P). Figure 8-1 shows the extended automaton. The description of the states and transitions follows immediately.

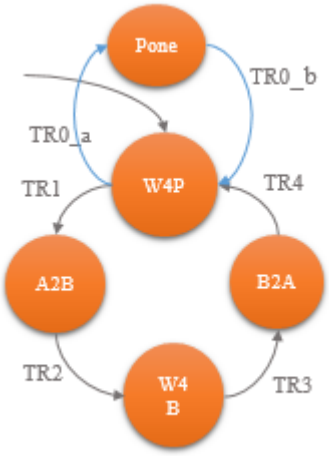

Figure 8-1: Multiple-grips robot automaton

States

$Pone$: pick one product

$W4P$: the robot is waiting for a product

$A2B$: the robot is moving from conveyor A to conveyor B

$W4B$: the robot is waiting for a box

$B2A$: the robot is moving from conveyor B to conveyor A

Transitions

TR0_a: ∃ p: $L^r_{pi,min} \leq x_p \leq L^r_{pi,max}$ **AND** p is assigned to robot r **AND** the number of picked products is lower than the number of grips

TR0_b: $Pone$ state has been active for more than T_p

TR1: number of picked products equals to the number of grips

TR2: A2B state has been active for more than $\frac{T_m}{2}$

TR3: ∃ b: $L^r_{pl,min} \leq x_b \leq L^r_{pl,max}$ **AND** b is assigned to robot r **AND** b is available (not full)

TR4: B2A state has been active for more than $\frac{T_m}{2}$

8.3 Model verification

Same validation procedure is applied to the extended model where the model is verified in terms of lost products and the time between two pickups for each robot. A cyclic strategy is adopted to control the flow of products. In this case study, model 3 is extended to model 4 where the assumption of one large product is removed. The following table shows the model parameter differences.

Table 12: Model parameter for both simplified and extended models

MODEL [#]	ROBOTS [#]	N^p_{row}	N^b_{layer}	N^r_{grips}	PR	V_{pi}	V_{pl}	T_m	T_p	Δx_b
3	6	2	4	1	120	30	13.5	1.46	-	0.45
4	6	2	4	5	600	30	13.5	0.96	0.1	0.45

In this case, the real counterpart of model 4 loses a number equal to 87 products whereas the extended model loses 85. Even though the numbers are not exactly the same the transient behaviour is similar on both cases. The plant behaviour in terms of the time between transferring two groups of products is plotted in Figure 8-3 and Figure 8-4. Comparing these two figures with the ones of chapter 3 that corresponds to model 3, the simplified model

resulted in a slightly different transient behaviour. The settling time, however, is around 180 second for the extended model whereas it is around 250 second for the simplified one.

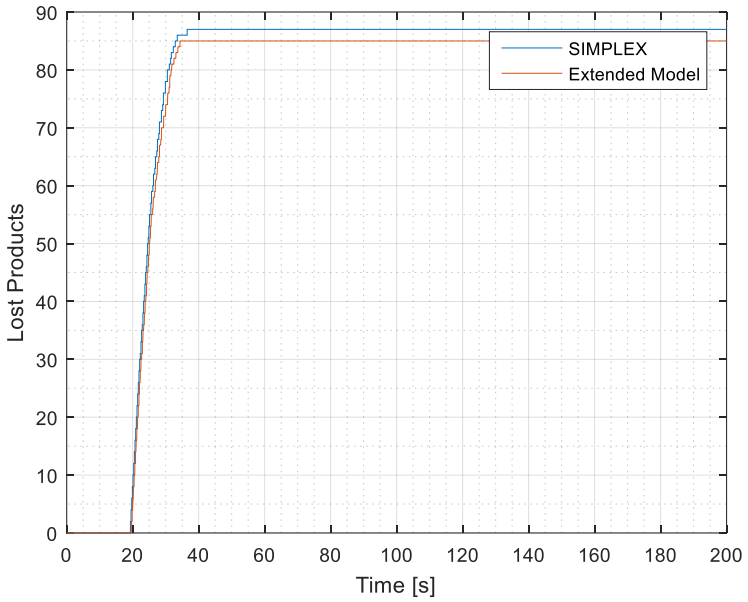

Figure 8-2: Validtaion in terms of the behaviour of the number of lost prodcuts

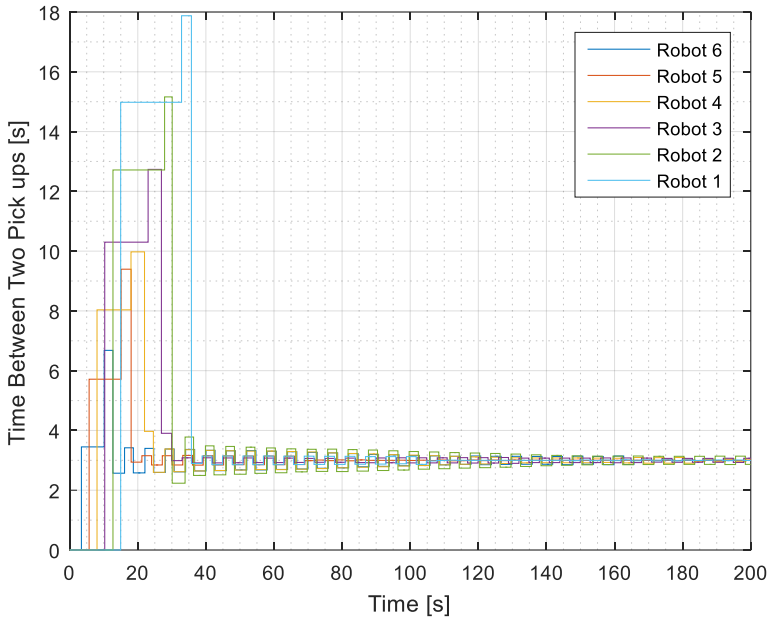

Figure 8-3 Time between transferring two groups of products for multi grip model 4 (RST)

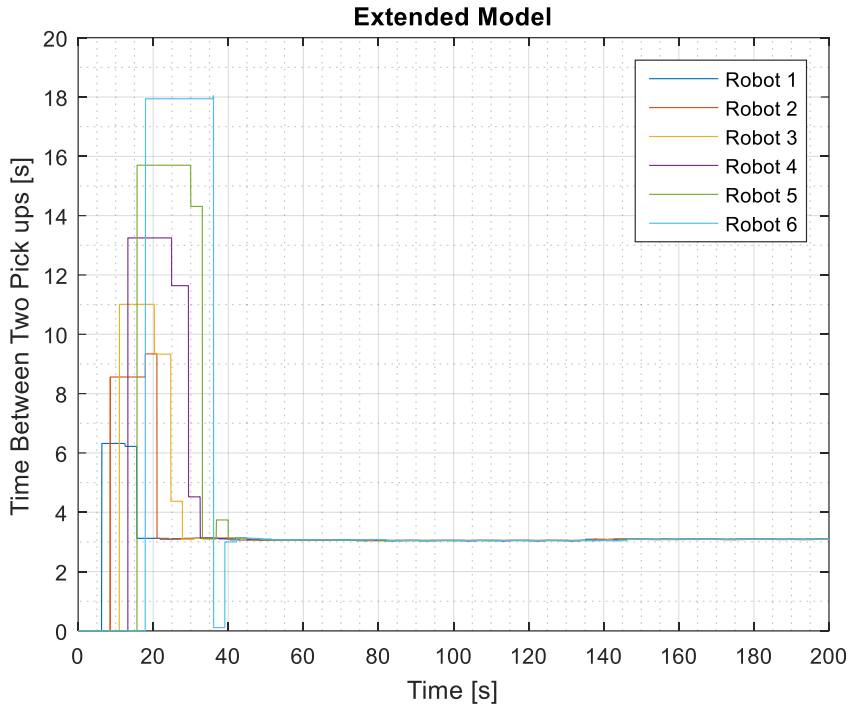

Figure 8-4: Time between transferring two groups of products for multi grip model 4 (Extended Model)

8.4 Control strategy

The plant is initially controlled with a cyclic assignment and led to a loss of products equals to 85. A number of improvements to this controller has been discussed in chapter 6. In Figure 8-5, the plant has been optimized by manipulating the initial boxes' distance (IBD). Unlike the results in chapter 6, the number of lost products does not drop down to zero. Instead, an IBD of 1.6 [m] reduces N_{loss}^{p} to 35. Following the proposed algorithm, the plant is initialized with an IBD of 1.6 [m] and the plant is optimized over the switching instance (SI). Each vertical line of Figure 8-6 corresponds to the number of loss products when the schedule is switched to a cyclic one after n products. N_{loss}^{p} is minimized to 9 when the first 170 products are assigned to robots in a greedy manner. Products with an identification number greater than 170 are assigned cyclically. The last piece of the algorithm is to increase the distance between boxes. The distance is first increased by 10[cm] which leads to Δx_b equals 55 [cm], consequently, the place conveyor velocity is updated to 16.5 [m/min]. There is no need to obtain with simulation another value of IBD. Equation (6.15) computes an estimate of the new IBD based on the old one. The last piece of the algorithm is to optimize over the SI. Following this procedure the number of lost products is minimized to zero. Figure 8-7 illustrates the decreasing number of lost products with respect to Δx_b equals to 55 [cm].

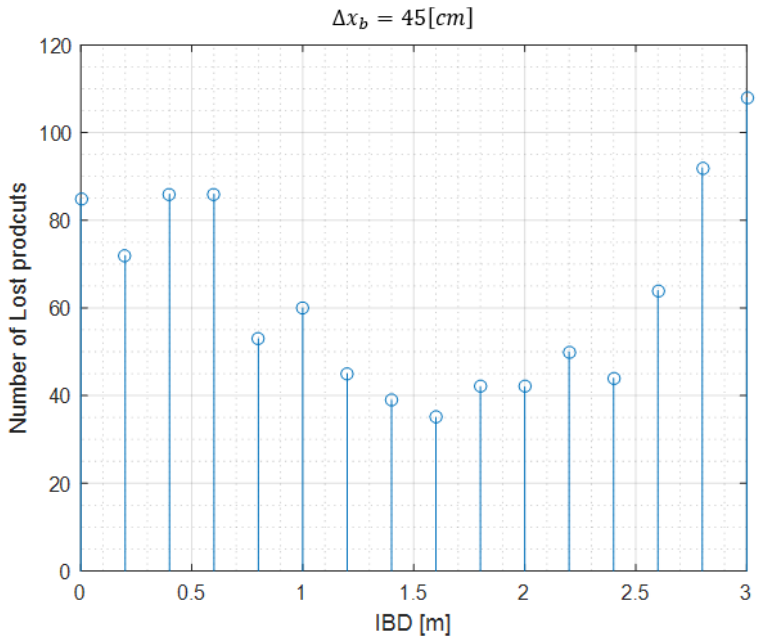

Figure 8-5: Optimization over IBD

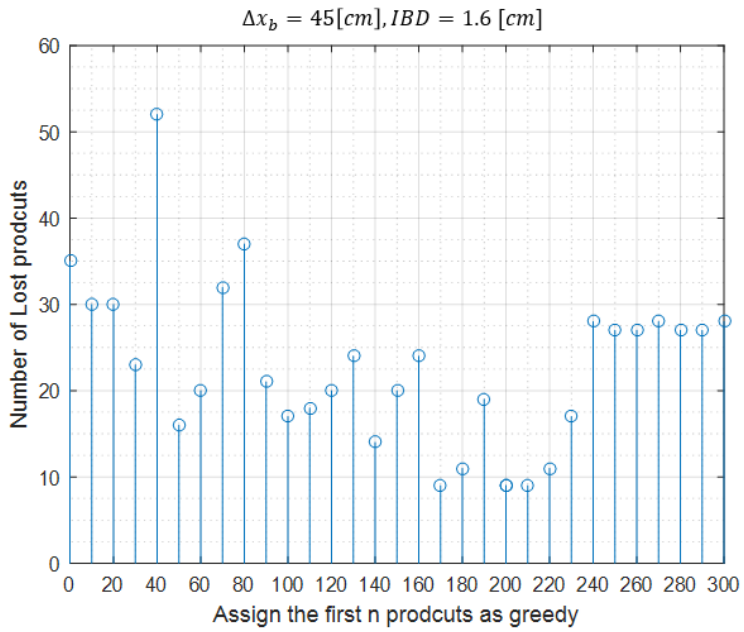

Figure 8-6: Optimization over the number of greedy products

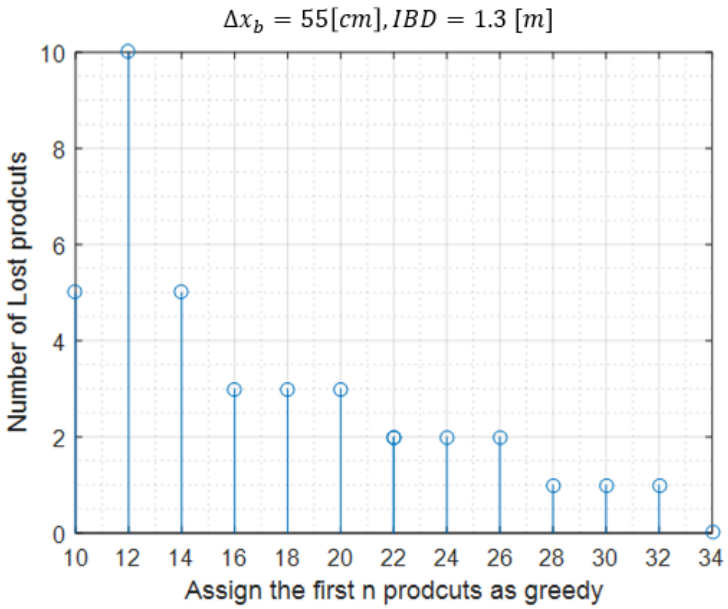

Figure 8-7: Optimizing over SI with delta Δx_b equals to 55[cm]

8.5 Conclusion

An extension to the simplified model is introduced to reflect the multi-grip robots. The extended model has been validated against RST in terms of lost products and the time between transferring two groups of products. Finally, following the number of improvements, stated in chapter 6, to the cyclic assignment, the number of lost products is minimized.

9 Conclusions and final remarks

The problem of modelling and scheduling of a packaging line in the food industry was presented. The particular plant that has been studied is one that consists of two parallel conveying belts and a number of robotic manipulators mounted in a series configuration. One belt transfers the incoming products and the other carries incoming boxes where products are packed. A simplified modelling of the system was proposed and verified in chapter 3. The scheduling problem have been studied into two different contexts. Chapters 4 and 5, proposed the scheduling problem as a mixed integer linear program (MILP) optimization problem. The solution of which is applied to the plant at each occurrence of significant events. Chapter 6 tackled the problem under a simulation based optimization (SbO) technique where a number of improvements is applied to a steady state solution, namely the cyclic assignment. The model in chapter 8 has been extended to multi-grip robotic manipulators and SbO is used to find the proper assignment.

It has been shown that MILP allowed the scheduling problem to be modelled in a closed loop configuration which superior in terms of acting in an online manner to real-time disturbances. However, the current status of the proposed solution is computationally heavy which makes the solution infeasible for such online configuration. Although a computationally improved solution was shown, it is still not fast enough.

For the future work, instead of assuming a single velocity variable, a future velocity trend along the prediction horizon must be considered. This essentially might lead to a better estimation of the velocity profile and, as a consequence, the assignment of products and boxes would reach a uniform distribution among robots. Implementation of such solution requires the reduction of the computational cost which could be done through a modification of the products' decision variables. Assuming that the decision of picking products along the vertical axis is the same, these products could be assumed to belong to a virtual block represented with a single decision variable. This means that the computational cost is independent of the number of the rows of products. Finally, the simulation based optimization is an offline solution that could be applied to minimize the number of lost product in the start-up transient on a generic case of co-flux plant layouts. Since it is an offline solution, this technique inherits the problems of open loop solutions when disturbances occur, such as severe variations of production rate. So, for the future work, further investigations on online disturbances could be performed.

References

[1] L. A. Wolsey, Integer Programming, John Wiley & Sons, Inc., 1989.

[2] E. F. Camacho and C. Bordons Alba, Model Predictive Control, Springer, 2007.

[3] G. Ferrari, L. Ferrarini, A. Petretti and E. Pizzi, “Modeling and design of an optimal line manager of a packaging system with MILP,” in *IEEE*, 2015.

[4] R. Mattone, M. Divona and A. Wolf, "Sorting of items on a moving conveyour belt. Part2: performace evaluatino and optimization of pick-and-place operations," Robotics and Computer Integrated Manufacturing, 2000.

[5] B. P. Gerkey and M. J. Mataric´, “A formal analysis and taxonomy of task allocation in multi-robot systems,” International Journal of Robotics Research, 2004.

[6] S. Dauod, H. Chehade, F. Yalaoui and L. Amodeo, “Efficient metaheuristics for pick and place robotic systems optimization,” 2014.

[7] M. Dorigo and T. Stützle, Ant Colony Optimization, The MIT Press, 2004.

[8] B. Kalyanasundaram and K. Pruhs, “Online Weighted Matching,” Journal of Algorithms, 1993.

[9] H. W. Kuhn, “The Hungarian Method for the Assignment Problem,” Naval Research Logistics Quarterly , 1955.

[10] H. I. Bozma and M. E. Kalalioglu, “Multirobot Coordination in Pick-and-Place tasks on a Moving Conveyor,” Robotics and Computer-Integrated Manufacturing, 2012.

[11] L. Liu and D. A. Shell, “Optimal Market-based Multi-Robot Task Allocation”.

[12] D. M. Bertsekas, “The auction algorithm for assignment and other network flow problems,” Scholarly Publications, US, 1990.

[13] B. Dias, N. Kalra, R. Ziot and A. Stentz, “Market-Based Multirobot Coordination: A Survey and Analysis,” IEEE, 2006.