POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea Specialistica in
Ingegneria Meccanica

SIMULAZIONE E OTTIMIZZAZIONE DI SISTEMI PULL MEDIANTE PROGRAMMAZIONE MATEMATICA

Relatore: Prof. Andrea MATTA

Correlatore: Dott. Giulia PEDRIELLI

Tesi di Laurea di:

Emanuele VERCESI
Matricola: 725690

Anno Accademico 2012 - 2013.

Ringraziamenti

Giunto alla conclusione di questo lavoro voglio formulare alcuni ringraziamenti: innanzitutto ai miei genitori e a mia sorella, che mi hanno sostenuto e accompagnato in questi anni di Università .

Ringrazio la mia fidanzata Laura, per tutto l'appoggio e la forza che mi ha dato, i miei amici, i miei compagni di Università: E. Ogliari, F. Serra, A. Stringa e tutti gli altri. Un sentito grazie anche a chi, per brevità, non è stato menzionato.

Ringrazio l'Ing. Giulia Pedrielli che in questi mesi di sperimentazioni mi ha sempre aiutato e consigliato al meglio delle sue possibilità.

Ringrazio la Professoressa Arianna Alfieri per l'attenzione e la disponibilità che ha dimostrato nei miei confronti.

Un ringraziamento, infine, va al Professor Andrea Matta per l'attenzione e la disponibilità con cui mi ha seguito durante il lavoro di preparazione della tesi.

INDICE

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

Sommario

L’ottimizzazione di sistemi produttivi rappresenta un problema complesso.
La ricerca delle condizioni operative ottime implica l’analisi delle caratteristiche e delle prestazioni dei sistemi stessi.
In altre parole per risolvere un problema di ottimizzazione è necessario anche risolverne uno di simulazione.
In questo lavoro di tesi, ci si concentra sull’ottimizzazione di linee di produzione gestite mediante politiche di controllo di tipo Pull. I sistemi gestiti mediante queste politiche prendono il nome di sistemi Pull e sono caratterizzati dal fatto che la produzione è avviata dalla domanda di parti finite. La famiglia di logiche di tipo Pull consta di diverse tecniche sviluppate negli anni. Nel presente lavoro verranno prese in esame due di tali politiche (e i risultanti sistemi di produzione): Kanban Control System (KCS) e Base Stock Control System (BSCS).
Le principali misure di prestazione che vengono influenzate dall’adozione di queste logiche sono il livello di servizio, inteso come la percentuale di domanda soddisfatta senza ritardo e il livello di work in process (WIP) presente nel sistema. Diverse logiche hanno un effetto diverso su questi indicatori: i sistemi KCS controllano il livello di WIP mentre l’adozione di un sistema BSCS porta all’aumento della reattività della domanda, fattore estremamente rilevante nel caso di domanda particolarmente variabile, tuttavia non viene controllato il livello di WIP generalmente superiore al caso del sistema KCS che, d’altra parte, realizza un livello del servizio inferiore.
In generale, l’implementazione di una precisa logica si traduce nello stabilire il valore da assegnare ai valori che caratterizzano la politica stessa, ossia i *parametri di controllo*.
La logica kanban è caratterizzata da un unico parametro di controllo, ossia il numero di risorse cartellino Kanban. Queste rappresentano il numero di autorizzazioni presenti ad ogni stadio e vengono utilizzate per vincolare il rilascio delle parti da uno stadio di produzione al successivo. Ne risulta che il numero delle parti presenti tra due stadi di produzione ad ogni istante di tempo è limitato superiormente.
Anche la logica Base Stock è caratterizzata da un unico parametro: il livello iniziale del magazzino ad ogni stadio della produzione. Il sistema viene poi gestito in modo che, ad ogni istante di tempo, il numero delle parti disponibili nel magazzino di ogni stadio sommato al numero delle parti richieste allo stadio a monte (*inventory position*) assuma lo stesso valore del livello iniziale delle scorte che prende, appunto, il nome di *base stock level*. Diverse sono le varianti nei problemi di ottimizzazione per sistemi KCS o BSCS. Tipicamente si analizza il trade-off tra il costo associato al livello di kanban e il costo di

backlog (costo associato al numero di parti in ritardo). Altri modelli di ottimizzazione, imponendo un certo livello di servizio, minimizzano il costo associato ad alcuni indicatori delle prestazioni (per esempio livello di WIP) al variare della configurazione del livello di Kanban o Base Stock. Alcuni casi, minimizzano una funzione di costo associata al livello di Kanban o Base Stock necessario per garantire un dato livello di servizio.
Esistono numerosi metodi per risolvere problemi di ottimizzazione. Le tecniche maggiormente utilizzate in letteratura sono le euristiche, programmazione matematica (MPR), tecniche meta-euristiche e modelli di ottimizzazione iterativi basati su simulazione (Response Surface Methodology - RSM).
Nella fase di simulazione si possono valutare le prestazioni del sistema, in quanto il modello di simulazione incorpora la descrizione (implicita o esplicita) del comportamento dello stesso. Le principali tecniche di simulazione impiegate sono: la simulazione ad eventi discreti (DES) e i metodi analitici basati su reti di code e programmazione matematica.
Nei modelli di programmazione matematica, la dinamica del sistema viene rappresentata attraverso una serie di vincoli; la funzione obbiettivo viene utilizzata per guidare la soluzione verso la rappresentazione della traiettoria del sistema. La modellizzazione MPR, permette di integrare la fase di simulazione e ottimizzazione. In questo approccio, la fase di ottimizzazione contiene già le informazioni relative alla dinamica del sistema, evitando quindi la necessità di testare la soluzione ottima attraverso una fase di simulazione. In questo modo il sistema produttivo è ottimizzato e simulato allo stesso tempo.
Il problema derivante da questo tipo di tecnica è legato alla risoluzione di un modello di programmazione lineare intera. Tuttavia sono stati proposti in letteratura delle tecniche per approssimare questi modelli interi che preservano la linearità dei modelli matematici anche quando sono utilizzati per la simulazione.
In questo lavoro di Tesi, partendo dall'approccio di simulazione mediante Programmazione Lineare (MRP), è stato sviluppato un modello approssimato che permette l'ottimizzazione dei livelli di controllo dei sistemi KCS e BSCS.
Il modello presentato, attraverso la presenza della variabile continua *Time Buffer*, oltre a simulare il comportamento del sistema KCS o BSCS, permette di ottenere il livello di Kanban e Base Stock ottimizzato.
Il concetto di Time Buffer deriva da un'approssimazione continua che mira alla sostituzione delle variabili decisionali intere definite nel modello di ottimizzazione originale. In generale, per approssimare una variabile discreta con un Time Buffer, deve essere possibile descrivere formalmente i suoi effetti sugli eventi che caratterizzano la dinamica del sistema produttivo.
Attraverso la risoluzione del modello con i Time Buffer è possibile ricavare buoni risultati in termini di soluzione intera approssimata e robusti limiti relativi alla soluzione esatta (limite inferiore e superiore all'interno dei quali è contenuta la soluzione esatta).

I risultati ottenuti attraverso la soluzione del modello approssimato con la variabile time Buffer, sono stati confrontati con quelli derivanti da modelli di simulazione e ottimizzazione realizzati mediante software dedicati.

In sintesi, questa tesi fornisce un nuovo approccio nel campo dell'ottimizzazione dei livelli di controllo per sistemi KCS e BSCS, utilizzando la programmazione matematica. I modelli ottenuti possono essere sfruttati come primo passo in una procedura di ottimizzazione, in quanto forniscono un algoritmo in grado di offrire una buona soluzione approssimata e robusti limiti alla soluzione esatta.

Parole chiave: Kanban, Base Stock, Sistemi Pull, Modelli di Ottimizzazione e Simulazione, Programmazione Matematica.

Abstract

The optimization of production systems is a complex problem.
The search for the optimal operating conditions involves the analysis of the characteristics and performance of these systems.
In other words, to solve an optimization problem is also necessary to solve one simulation.
In this dissertation work, we focus on the optimization of production lines managed by policy type control Pull. The systems managed by these policies take the name of Pull systems and are characterized by the fact that the production is started consequently the request of finished parts. The family of logics Pull consists of several techniques developed over the years. In the present work will be considered two such policies (and the resulting production systems): Kanban Control System (KCS) and Base Stock Control System (BSCS).
The main measures of performance are affected by the adoption of these logics are the service level, defined as the percentage of demand satisfied without delay and the level of work in process (WIP). Several logics have a different effect on these indicators: KCS control the WIP level while the adoption of BSCS leads to increased demand reactivity, that is extremely important in the case of particularly variable demand, however, is not checked the level of WIP generally higher than the case of the KCS system that, on the other hand, provides a lower service level.
The implementation of a specific Pull logic translates in establishing the importance to assign to the values that characterize the policy, ie, the control parameters.
The kanban logic is characterized by a single control parameter, the number of resources Kanban card. These represent the number of authorizations present at every stage and are used to make the issue of the parts from one production stage to the next. It follows that the number of parts present between two production stages, at each instant of time is upper bounded.
Even Base Stock logic is characterized by a single parameter: the initial inventory level at each stage of production. The system is then operated so that, at each instant of time, the number of parts available in the warehouse of each stage plus the number of parts required to upstream (inventory position) assumes the same value of the initial inventory level taking, the name, in fact, of base stock level. There are several variations in the optimization problems for KCS or BSCS. Typically analyzing the trade-off between the cost associated with the level of kanban and the cost of backlog (as cost associated to the number of parts in delay). Other optimization models, requiring a certain level of service, minimize the cost associated with some performance indicators (for example, level of WIP) to vary the configuration of the level of kanban or Base Stock.

Some cases, minimize a function cost associated with the level of Kanban or Base Stock necessary to provide a particular service level.
There are several methods for solving optimization problems. The techniques most commonly used in the literature are heuristics, mathematical programming representation (MPR), meta-heuristics technical and optimization models based on iterative simulation (Response Surface Methodology - RSM).
In the simulation phase can evaluate the performance of the system, as the simulation model incorporates the description (implicit or explicit) of the system behavior. The main simulation techniques used are: the discrete event simulation (DES) and the analytical methods based on networks of queues and mathematical programming.
In mathematical programming models, the dynamics of the system is represented by a set of constraints, the objective function is used to guide the solution towards the representation of the trajectory of the system. The MRP representation allows to integrate the phase of simulation and optimization. In this approach, the optimization phase already contains the information relating to the dynamics of the system, thus avoiding the need to test the optimal solution through a simulation phase. In this way, the production system is optimized and simulated at the same time.
The problem arising from this kind of technique is related to the resolution of a integer linear programming model. However in literature have been proposed several techniques to approximate these integers models that preserve in linearity for the mathematical models even when they are used for the simulation.
Starting from the simulation approach using Mathematical Programming Representation (MRP), in this dissertation work was developed a model that allows the optimized control levels for KCS and BSCS.
The presented model, through the presence of the continuous variable Time Buffer, in addition to simulate the behavior of Kanban Control System or Base Stock Control System, allows to obtain the level of Kanban and Base Stock optimized.
The concept of Time Buffer is derived from an approximation continues that aims to replace the integer decision variables defined in the original model of optimization. In general, to approximate a discrete variable with a Time Buffer, it must be possible to formally describe its effects on the events that characterize the dynamics of the production system.
Through the resolution of the model with the Time Buffer is possible to obtain good results in terms of the integer approximate solution and robust approximate bound for the exact solution (upper and lower bound within which the exact solution is contained).
The results obtained through the solution of the approximate model with Tim Buffer variable, were compared with those derived from models of simulation and optimization made using dedicated software.

In summary, this thesis provides a new approach in the field of control levels optimization for KCS and BSCS systems, using Mathematical Programming Representation. The models obtained can be exploited as a first step in an optimization process, as they provide an algorithm able to offer a good approximate solution and robust bounds for the exact solution.

Keywords: Kanban, Base Stock, Pull Systems, Optimization and Simulation Models, Mathematical Programming Representation.

1 Introduzione

La gestione dei sistemi produttivi, a partire dagli ultimi vent'anni del secolo scorso, ha cominciato a focalizzare l'attenzione su di una nuova realtà produttiva: la Lean Production.
A partire dagli anni 80', gli studiosi focalizzarono la loro attenzione sul comportamento di una logica produttiva presentata da Toyota. Questo tipo di approccio, risulta coordinato da una serie di metodi che si basano sulla filosofia del Just In Time (JIT). Il principio base di questa è basato sull'eliminazione degli sprechi e sul continuo miglioramento della produttività.
Questa nuova filosofia ha dato origine a una serie di tecniche di gestione della produzione che prendono il nome di politiche Pull.
La produzione nei sistemi che adottano queste politiche è "tirata" dal mercato, cioè dalla domanda effettiva del cliente.
Le prime logiche Pull sviluppatesi sono: Kanban Control System (KCS) e Base Stock Control System (BSCS). Queste tecniche sono applicate a sistemi produttivi caratterizzati da numerosi stadi della produzione dove con essi si intendono le varie fasi attraverso le quali le parti vengono lavorate.
Le tecniche Pull sono definite da parametri di controllo e la configurazione stessa di questi ultimi, influisce sulle performance del sistema al quale sono applicate le logiche produttive.
Per i sistemi KCS il parametro di controllo è rappresentato dal numero di cartellini Kanban che vengono messi a disposizione di ogni stadio della produzione. Questi rappresentano il numero di autorizzazioni presenti ad ogni stadio e vengono utilizzate per vincolare il rilascio delle parti da uno stadio di produzione al successivo. Ne risulta che il numero delle parti presenti tra due stadi di produzione ad ogni istante di tempo è limitato superiormente.
Anche la logica Base Stock è caratterizzata da un unico parametro: il livello iniziale del magazzino ad ogni stadio della produzione. Il sistema viene poi gestito in modo che, ad ogni istante di tempo, il numero delle parti disponibili nel magazzino di ogni stadio sommato al numero delle parti richieste allo stadio a monte (*inventory position*) assuma lo stesso valore del livello iniziale delle scorte che prende, appunto, il nome di *base stock level*.
Attraverso queste due tecniche è possibile migliorare le prestazioni della linea produttiva in termini di livello di servizio inteso come la capacità di soddisfare senza ritardi, una certa percentuale della domanda. Le tecniche KCS e BSCS permettono al sistema di aumentare la reattività di risposta in caso di domanda variabile di parti (BSCS) e di migliorare il livello di work-in-process (KCS).
Per ottenere performance ottime è fondamentale la conoscenza dei livelli ottimi di Kanban e Base Stock.

1.1 Obiettivo della Tesi

In questo lavoro di tesi, ci si concentra sull'ottimizzazione di linee di produzione gestite mediante politiche di controllo di tipo Pull. Le tecniche Pull prese in considerazione sono Kanban Control System (KCS) e Base Stock Control System (BSCS).
Attraverso queste tecniche, è possibile migliorare le prestazioni della linea produttiva in termini di livello di servizio, inteso come la percentuale di domanda soddisfatta senza ritardi. Le performance dei sistemi che operano con tecniche Pull sono influenzate dal valore dei parametri di controllo delle politiche stesse. Per ottenere prestazioni ottime dai sistemi è necessario determinare il livello ottimo dei parametri di controllo.
L'ottimizzazione di sistemi produttivi rappresenta un problema complesso e di notevole interesse. La ricerca delle condizioni operative ottime, nei sistemi manifatturieri, necessita di un' accurata analisi delle caratteristiche e delle prestazioni dei sistemi stessi. Da qui la necessità di affrontare due differenti problemi: uno di simulazione e uno di ottimizzazione.
In questa analisi, partendo da un approccio di simulazione mediante programmazione lineare esatta (MPR), è stato sviluppato un modello approssimato che permette l'ottimizzazione dei livelli di controllo dei sistemi KCS e BSCS.
L'ottimizzazione di questo modello approssimato si basa sulla minimizzazione del costo complessivo del livello dei parametri di controllo nel rispetto di un livello di servizio.
Il processo di ottimizzazione mediante modello MPR, oltre a simulare il comportamento del sistema di produzione, consente di ottenere una configurazione ottima per i parametri di controllo della politica Pull in esame che permette al sistema di ottimizzare le proprie prestazioni in termini di work in process (KCS) o di risposta alle variazioni della domanda di parti finite (BSCS).

1.2 Stato dell'arte

Per comprendere le problematiche relative all'ottimizzazione dei parametri di controllo di politiche Pull applicate a linee di produzione, occorre prima focalizzare l'attenzione sul comportamento dei sistemi Pull in generale.
Solo dopo avere compreso la dinamica dei sistemi gestiti con queste tecniche, è possibile concentrarsi sulla simulazione e l'ottimizzazione delle Politiche Pull stesse.

1.2.1 Sistemi di tipo Pull

Con meccanismo di tipo Pull, si intende un sistema produttivo la cui produzione, a partire dall'ultima stazione di lavoro, viene "tirata" dalla domanda del cliente finale.
Esistono numerose tecniche di gestione di tipo Pull: Kanban Control System (KCS), Base Stock Control System (BSCS), CONWIP, Generalized Kanban Control System (GKCS) ed Extended Kanban Control System (EKCS).
La politica Kanban Control System (KCS) è caratterizzata da un singolo parametro di controllo che vale per tutti i prodotti e per tutti gli stage; questo parametro è il numero di cartellini Kanban. Questi rappresentano il numero di autorizzazioni presenti ad ogni stadio e vengono utilizzate per vincolare il rilascio delle parti da uno stadio di produzione al successivo. Ne risulta che il numero delle parti presenti tra due stadi di produzione ad ogni istante di tempo è limitato superiormente, infatti, questa tecnica è in grado di limitare il livello di work-in-process (WIP) e di controllare il numero di parti finte ad ogni stage.
Anche la politica Base Stock Control System (BSCS) possiede un unico parametro di controllo rappresentato dal numero di posizioni a stock o base stock level; ossia il livello iniziale del magazzino, per ogni istante, ad ogni stadio della produzione. Il vantaggio fondamentale ottenuto dall'adozione del BSCS è una maggiore reattività di risposta in caso di domanda variabile di parti. Lo svantaggio di questa tecnica è caratterizzato dal fatto che il WIP non è controllato.

1.2.2 Ottimizzazione di Sistemi Pull

Dopo aver compreso il comportamento delle politiche di tipo Pull, si può procedere all'ottimizzazione dei livelli di controllo delle politiche stesse.
Numerosi modelli di ottimizzazione sono stati sviluppati per quanto riguarda i sistemi KCS e BSCS. Sono state sviluppate tecniche euristiche, di programmazione matematica (MPR), tecniche meta-euristiche e modelli di ottimizzazione iterativi basati su simulazione (Response Surface Methodology)
Gupta e Al-Turki [1] hanno introdotto una metodologia sistematica per ottimizzare il livello di Kanban in un sistema JIT. Il modello di ottimizzazione proposto minimizza il livello di WIP e il costo di backlog.
Sarker e Balan [2] hanno ottimizzato il numero di kanban richiesti per il trasferimento di parti tra due stazioni; questo modello è valido per sistemi single stage e per sistemi multi stage.
In alcuni modelli De Araujo [3], l'ottimizzazione tiene in considerazione come unico vincolo il costo di backlog ossia il costo che si manifesta quando una parte giunge in ritardo. In questo modello viene introdotta una funzione di costo totale dipendente dal costo di backlog complessivo.
Ulteriori modelli per l'ottimizzazione del livello di kanban si servono di: Reti di Code, Reti di Petri, Modelli Genetici e Modelli Matematici.

Per quanto riguarda l'ottimizzazione del livello di Base Stock, i primi studi furono di Clark e Scaarf [4]. Essi sono riusciti a stabilire che un sistema produttivo seriale potesse essere schematizzato come una serie di singole unità. Hanno preso in considerazione tempi discreti, un orizzonte temporale finito e una domanda distribuita esponenzialmente. Hanno sviluppato un modello di ottimizzazione esatta e l'algoritmo per poterlo risolvere.
Federgruen e Zipkin in [5] hanno adattato i risultati, ottenuti con l'algoritmo di Clark e Scaarf, all'orizzonte infinto e semplificando l'algoritmo iniziale.
Rosling [6], Lagenhoff e Zijm [7] continuarono a sviluppare i risultati attraverso un'analisi numerica.
Chen e Zheng [8] in hanno rielaborato i risultati presenti in letteratura riuscendo a fornire una versione dell'algoritmo in dominio di continuità.
Vista l'onerosità in termini di sforzo computazionale di questi modelli di ottimizzazione per BSCS, negli ultimi anni sono comparsi numerosi modelli euristici. Essi consentono il raggiungimento di risultati paragonabili a quelli ottenuti con il modello esatto, con un risparmio sia di tempo che di sforzo computazionale impiegato.
In ognuno di questi modelli euristici viene presentata un funzione di costo, dipendente dal valore del livello di base stock. L'ottimizzazione avviene minimizzando la funzione di costo per un data configurazione del parametro di controllo sopracitato. Molto spesso i risultati di questi modelli euristici sono influenzati dal tipo di domanda che caratterizza il sistema. Per alcune distribuzioni statistiche della domanda, non forniscono buoni risultati.
In numerosi contributi Zjim e Van Houtum [9] hanno confrontato i risultati di questi modelli euristici con sistemi di previsione della produzione come il Material Requirement Planning (MRP).
In altri modelli euristici come in Gallego e Zipkin [10] è definita una funzione di costo, associata al livello di base stock di ogni stadio della produzione, che viene minimizzata seguendo un algoritmo da loro stessi introdotto.
Nella trattazione di Shang and Song [11], viene sviluppato un metodo euristico per l'ottimizzazione del livello di inventory in sistemi multi-stage. Il contributo del loro lavoro offre un vantaggio sia computazionale che di implementazione. Attraverso questo modello euristico, per ogni stage del sistema produttivo, vengono ricavate due funzioni che limitano superiormente e inferiormente la reale funzione di costo complessivo dell'inventory. La funzione di costo è definita al variare del livello di base stock. La minimizzazione di queste due funzioni limite, definisce il valore di inventory minimo e massimo per ogni stage.

1.2.3 Approccio mediante Mathematical Programming Representation (MPR)

L'ottimizzazione di sistemi produttivi rappresenta un problema complesso e di notevole interesse. La ricerca delle condizioni operative ottime, nei sistemi manifatturieri, necessita di un' accurata analisi delle caratteristiche e delle prestazioni dei sistemi stessi. Da qui la necessità di affrontare due differenti problemi: uno di simulazione e uno di ottimizzazione.
Un contributo fondamentale per l'accorpamento della fase di simulazione e ottimizzazione delle condizioni operative dei sistemi produttivi, è stato apportato da Chan e Schruben [12] attraverso lo sviluppo di modelli di programmazione lineare (MPR).
Infatti, se per risolvere la fase di simulazione viene adottato un modello di programmazione matematica MPR, essa stessa risulterà estremamente legata ad un problema di ottimizzazione vincolata.
L'idea alla base dei modelli MPR contempla la definizione di un set di vincoli che descrivono la dinamica del sistema e il loro successivo inserimento all'interno di un modello di ottimizzazione.
Questo significa che la bontà della soluzione data dall'ottimizzatore non deve essere controllata da un simulatore esterno, in quanto il modello di ottimizzazione possiede già al suo interno le regole che descrivono le dinamiche del sistema. In altre parole, il modello matematico ottimizza il sistema mentre lo sta simulando, Matta [13].
L'utilizzo della programmazione matematica per simulazione e ottimizzazione stabilisce un metodo alternativo nella studio dei problemi di simulazione-ottimizzazione permettendo di spostare l'attenzione da approcci di simulazione iterativi a modelli sequenziali di ottimizzazione.
La modellizzazione mediante programmazione lineare è stata applicata anche a contesti specifici come i sistemi Pull. In Matta, Alfieri [14] è stato proposto un modello di programmazione lineare delle principali politiche Pull applicabile a sistemi manifatturieri multi-stage e monoprodotto. Questi modelli se inizializzati con gli stessi parametri di un modello di simulazione, permettono di ottenere gli stessi risultati in quanto simulano la stessa sequenza di eventi.
Questo tipo di modellizzazione mediante programmazione Lineare, ha numerosi vantaggi. Il più grande di questi è rappresentato dal fatto che il sistema di produzione, che opera sotto una data politica di controllo, può essere rappresentato da un insieme di equazioni lineari (per esempio la f. obiettivo e i vincoli). Questo significa anche avere un vantaggio nello sviluppo della modellazione e soprattutto nella verifica del comportamento del modello stesso.

1.3 Contributo della tesi

In questa Tesi, ci si è basati sui concetti sviluppati nel modello di Programmazione Lineare (MPR) sviluppato per sistemi Pull in Matta, Alfieri

[14]. Come accennato nella sezione 1.2.3, il modello permette di simulare il comportamento dei sistemi secondo le politiche di gestione di tipo Pull. Questo approccio però, non consente di ottenere i valori ottimi relativi ai parametri di controllo che caratterizzano le politiche stesse.
In questo lavoro, basandosi sui modelli di simulazione sopracitati, sono stati introdotti due modelli di ottimizzazione che permettono di calcolare il valore ottimo dei livelli di Kanban e Base Stock per KCS e BSCS rispettivamente.
Partendo dall'approccio di simulazione mediante Programmazione Lineare (MRP), è stato sviluppato un modello approssimato che permette l'ottimizzazione dei parametri di controllo dei sistemi KCS e BSCS.
Il modello presentato, attraverso la presenza della variabile continua *Time Buffer*, oltre a simulare il comportamento del sistema KCS o BSCS, permette di ottenere il livello di Kanban e Base Stock ottimizzato.
Il concetto di Time Buffer deriva da un'approssimazione continua che mira alla sostituzione delle variabili decisionali intere definite nel modello di ottimizzazione originale. Attraverso la risoluzione del modello con i Time Buffer è possibile ricavare buoni risultati in termini di soluzione intera approssimata e limiti robusti relativi alla soluzione esatta (limite inferiore e superiore all'interno dei quali è contenuta la soluzione esatta).
I risultati ottenuti attraverso la soluzione del modello approssimato con la variabile time Buffer, sono stati confrontati con quelli derivanti da modelli di simulazione e ottimizzazione realizzati mediante software dedicati.

All'interno del Capitolo 2 si trova lo stato dell'arte che riguarda la Lean production, la classificazione delle principali politiche Pull, le tecniche di simulazione e ottimizzazione presenti in letteratura e l'introduzione all'approccio mediante Programmazione Lineare MPR.
Nel Capitolo 3 viene presentato il modello di Programmazione Lineare Approssimato coi Time Buffer, vengono definiti i limiti alla soluzione esatta (bounds) e si definisce un algoritmo per ottenere i risultati.
All'interno del Capitolo 4 si trovano i risultati relativi ai valori ottimi dei parametri di controllo per le politiche KCS e BSCS ottenuti con l'ottimizzazione approssimata. Questi ultimi, nella seconda parte del capitolo, sono stati confrontati con i risultati derivanti da un modello di ottimizzazione esatta realizzato mediante software dedicato.
Nel Capitolo 5 sono inseriti i commenti sui risultati e i possibili sviluppi futuri di questo lavoro di tesi.

2 Stato dell'arte

2.1 La politica di gestione Lean

Durante il secolo scorso, l'incertezza dei mercati e una domanda turbolenta ed irregolare hanno reso molto difficile la gestione della produzione in campo manifatturiero. Le previsioni sulla domanda stessa, sono diventate difficili e inaffidabili e rendono un sistema basato sulla previsione di essa molto difficile da attuare.
In questa realtà, assecondare le variazioni della domanda mantenendo un elevato livello di efficienza produttiva ha creato la necessità di nuove politiche per la gestione della produzione.
La *produzione snella* (*lean production*) rappresenta una risposta a questa necessità. Con questo termine si identifica una filosofia industriale ispirata al Toyota Production System.
I principi di questa nuova filosofia si basano sull'eliminazione degli sprechi, sull'identificazione delle aspettative della clientela, sull'eliminazione degli stadi della produzione che non creano valore per il cliente finale, sul miglioramento continuo della produzione e sulla produzione *Just In Time* (JIT), ossia la produzione interamente sottoposta al manifestarsi della domanda.
La riduzione dei tempi, ottenibile con un lavoro sulla riorganizzazione del flusso, favorisce certamente l'adozione del modello di produzione tirata dagli ordini dei clienti, ma per realizzarla, sono necessari strumenti e logiche appropriate. I sistemi di controllo che si basano su questa nuova strategia di gestione della produzione "tirata" dal mercato, cioè dalla domanda effettiva del cliente, vengono definiti *pull*, in contrapposizione ai sistemi più tradizionali di tipo *push*, in cui la produzione viene "spinta" in avanti a partire dalle previsioni della domanda stessa.

2.2 Confronto tra tecniche Push e tecniche Pull

La produzione snella si pone in alternativa alla tradizionale tecnica push. I sistemi di controllo fondati sulla logica push si basano su previsioni della domanda: in questa tipologia di sistemi la produzione schedulata autorizza la produzione nelle stazioni di lavoro delle varie fasi.
Il meccanismo di controllo di tipo pull, invece, lavora sulla base di sulla domanda che si manifesta, piuttosto che su sue previsioni.
Nahmias in [15], adottata la più tradizionale delle definizioni tra sistemi Push e Pull: un sistema Pull è un sistema che muove gli items da uno stadio al

successivo solo quando è esso a richiederlo o a necessitarne, mentre in un sistema Push la pianificazione della produzione avviene in anticipo.
Karmakar e Zipkin, in [16] e [17], definiscono i sistemi Pull come linee in cui la produzione inizia in reazione al realizzarsi della domanda mentre il sistema push avvia la produzione in anticipo rispetto all'insorgere della domanda stessa.
Sintetizzando, dall'analisi critica della letteratura, i sistemi pull possono essere distinti nei seguenti aspetti:

- In un sistema Pull la produzione è triggerata dalla domanda che si manifesta mentre un sistema Push inizia la produzione per fare fronte ad una domanda futura.
- Un sistema Pull è un sistema che controlla esplicitamente il WIP presente in un sistema produttivo, mentre un sistema Push non ha limiti espliciti all'ammontare di WIP complessivo.

Dopo avere descritto brevemente le differenze tra sistemi Push/Pull vengono presentate le principali politiche di controllo di tipo Pull.

2.3 Tecniche Pull

Le tre politiche pull tradizionali sono rappresentate dalle tecniche Kanban Control System (KCS), CONWIP Control System (CCS) e Base Stock Control System (BSCS).

2.3.1 Notazione e terminologia adottata

Si consideri un sistema produttivo nel quale la produzione risulta concentrata in una linea composta da diversi stadi della produzione.
Con il termine stadio, stage, fase o stazione della produzione, si intende un determinato processo o tipo di lavorazione che viene eseguito su ciascuna parte.
In questo tipo di sistema produttivo è esercitata una logica di tipo Pull che coordina il rilascio e la domanda delle parti tra gli stadi.
Lo schema riportato in fig. 2.1 si riferisce ad un sistema manifatturiero composto da due fasi in serie [18].

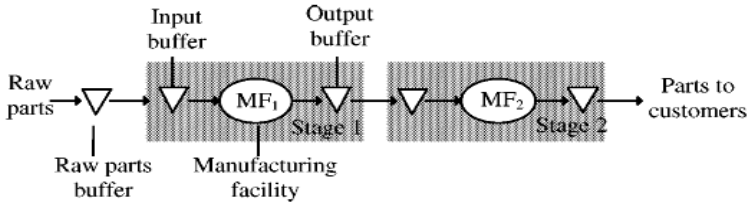

Figura 2.1. Sistema pull semplificato

Ogni stazione *i*-esima è indicata con MF_i.

All'inizio della linea è presente un magazzino di parti grezze (*Raw Parts Buffer*). Il magazzino di parti grezze fornisce l'alimentazione alla prima stazione della linea (MF_1). A monte e a valle di ogni stazione è presente un *buffer interoperazionale*, che disaccoppia stazioni contigue e ospita le parti in attesa di lavorazione (*Input Buffer* in fig. 2.1) o quelle che hanno terminato la fase di processamento (*Output Buffer* in fig. 2.1) [18].
Ogni buffer è caratterizzato da una capacità massima (finita o infinita). L'insieme di buffer di input, buffer di output e stazione manifatturiera, compongono il manufacturing facility (fig. 2.1). Ogni buffer di input è collegato con un buffer a monte. Alla fine della linea di produzione viene rappresentato lo stadio di domanda che riceve il prodotto finito generato dal sistema (*parts to customers* in fig. 2.1) [18].
Se la domanda non è evasa viene messa in attesa, in altre parole si assume che si possa avere domanda soddisfatta in ritardo e che tale ritardo non sia limitato (la domanda può attendere infinitamente). Questo ritardo nel soddisfacimento della domanda si denota con il termine *backlog*. Generalmente si associa un costo alla domanda soddisfatta in ritardo in modo da limitare l'incidenza di questo scenario.

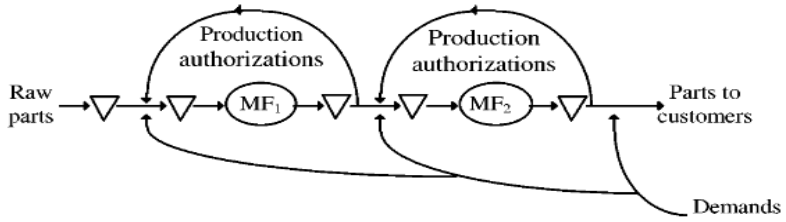

Figura 2.2. Rappresentazione dei flussi di un sistema Pull semplificato

Le principali entità che possono costituire un sistema pull (con opportune distinzioni a seconda della specifica politica adottata) sono (1) le parti da processare, (2) la domanda (3) le autorizzazioni all'avvio della produzione.
Le autorizzazioni, sono dei permessi che vengono inviati da una stazione di valle a una a monte per richiedere il rilascio di parti agli stadi a valle. Seguendo questa logica, la stazione MF_2 che necessita di parti grezze da lavorare, invierà alla stazione uno una un'autorizzazione per far si che essa possa cederle le parti di cui necessita.
Nella fig. 2.2, rispettando la trattazione in [18], vengono messi in luce i flussi riguardanti i tre elementi sopracitati.
Le frecce da monte verso valle indicano il flusso (delle parti, mentre tramite gli archi superiori modellano il flusso relativo al trasferimento della domanda agli stadi a monte. Gli archi rappresentano il flusso alle autorizzazioni.
Liberopoulos e Dallery in [18] propongono il metodo che utilizza le stazioni di sincronizzazione. Le stazioni di sincronizzazione consistono in un server con un

tempo di servizio istantaneo. Esso è caratterizzato da una serie di code in input e in output. La funzione delle stazioni di sincronizzazione è quella di accorpare due entità in input presenti in due code diverse e di restituire in output (su un'altra coda) una nuova entità frutto della sincronizzazione delle due in input.
Nei sistemi di controllo Pull le code di questo server possono contenere parti, domande, cartellini di autorizzazioni alla produzione o una combinazione di essi, come in [18].
Un esempio di stazione di sincronizzazione con tre code di input e due di output viene mostrato nella fig. 2.3.

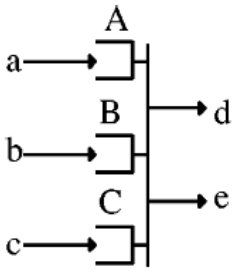

Figura 2.3. Stazione di sincronizzazione con tre input e due output

Riferendosi ad essa, si ipotizza che le code A e B contengano rispettivamente 4 e 3 clienti e che la coda C sia vuota. Appena arriva un cliente nella coda C, scatta il funzionamento della sincronizzazione. Tre clienti, quello di C, uno di A e uno di B, vengono fusi in due. In questo modo dalle code D ed E escono due clienti e nelle code A,B e C, restano 3, 2 e 0 clienti rispettivamente. Questo esempio può spiegare il comportamento delle autorizzazioni alla produzione. Il cliente della coda C rappresenta un'autorizzazione che giunge da valle. Essa viene sincronizzata con le parti presenti nella stazione B. Nella stazione A sono invece presenti le autorizzazioni da inviare a monte. Quindi l'arrivo sulla coda C di una autorizzazione da valle, permette il rilascio verso valle di una parte presente nella coda B. Contemporaneamente un' autorizzazione presente nella coda A viene inviata verso monte. In questo modo viene spiegato il funzionamento delle stazioni di sincronizzazione: per tre entità in input ci sono due entità in output [18].

2.3.2 Descrizione e confronto delle singole tecniche Pull

In questa sezione vengono presentati i modelli Pull più utilizzati nei sistemi produttivi di oggi. Le dinamiche di questi sistemi sono definite prendendo in considerazioni le nozioni contenute in [18] e riportate nella sezione 2.3.1.
I sistemi sono caratterizzati da code contenenti diverse entità: parti, domande o autorizzazioni. Lo scambio di queste entità viene eseguito mediante il meccanismo delle stazioni di sincronizzazione.

Seguendo la trattazione in [18], i sistemi ai quali viene applicata questa schematizzazione sono: (1) Kanban, (2) Base Stock, (3) CONWIP, (4) Generalized Kanban ed (5) Extended Kanban.
I primi tre sono caratterizzati da un unico livello di controllo, mentre gli ultimi tre ne posseggono due. Ognuno dei sistemi a due parametri, include, come caso particolare, entrambi i sistemi a parametro singolo. I parametri di controllo sono sempre K_i e S_i, ossia il numero di Kanban e il livello di Stock per ogni i-esimo stage.
Tutti i sistemi sono modellati come una rete di code con stazioni di sincronizzazione. Il nome delle code ed il loro contenuto risultano comuni in tutti i modelli pull analizzati [18].
La coda di tipo P_i contiene le parti finite dello stage i-esimo, la coda D_i contiene la domanda di parti finite allo stage i-esimo, la coda A_i contiene le autorizzazioni per la produzione relative allo stage i-esimo, la coda PA_i contiene una coppia formata da pezzi finiti e autorizzazioni alla produzione dello stage i-esimo, la coda DA_i contiene la coppia formata da domanda e autorizzazioni allo stage i-esimo, per i=1,..,N dove N è il numero degli stage [18].
La coda I_i contiene le parti che stanno aspettando di essere processate allo stage i-esimo, la coda P_0 rappresenta il buffer contenente le parti grezze, la coda D_{j+1} contiene le domande per la consegna delle parti finite ai clienti.
Le parti grezze relative al buffer P_0 riforniscono la prima stazione e di conseguenza tutta la linea di produzione. Alla stazione D_{N+1}, arriva la domanda di parti finite [18].
Definito questo schema a code, presentato in [18], generale per ogni sistema Pull, si passa nel dettaglio andando ad analizzare le principali politiche presenti in letteratura: KCS, BSCS, CONWIP, GKCS, EKCS.

2.3.2.1 Modello Kanban Control System – KCS

Numerosi sono gli studi presenti in letteratura relativi ai sistemi KCS (i riferimenti sono presenti in [19], [20], [21], [22], [23], [24] e [25]).
Un sistema KCS è caratterizzato dalla presenza di autorizzazioni (o cartellini) Kanban attaccati ad ogni parte circolante nel sistema. Essi permettono il trasferimento di parti o informazioni da uno strage al successivo.
Le autorizzazioni quindi, oltre a consentire la movimentazione della parti, servono a trasportare il flusso informativo della domanda dall'ultima stazione fino alla prima. Questo trasferimento delle informazioni legate alla domanda, avviene di stazione in stazione.
La fig. 2.4, che fa riferimento alla trattazione in [18], mostra un sistema KCS composto da due stage in serie.

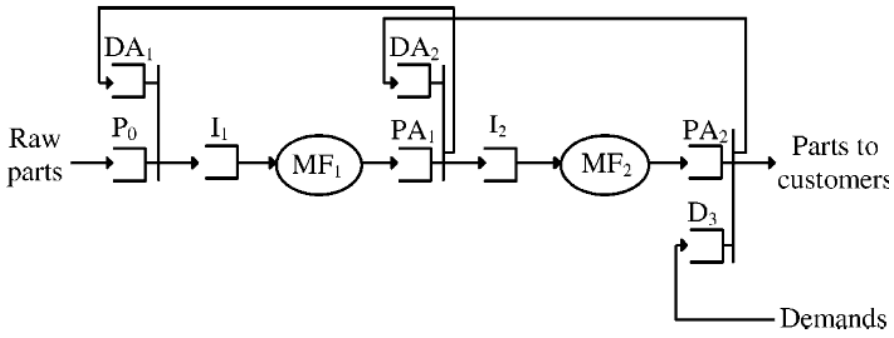

Figura 2.4. Sistema KCS con due stage in serie

Gli stage i-esimi i=1,2, sono caratterizzati da un numero K_i di autorizzazioni Kanban. La coda PA_i rappresenta il buffer di uscita dello stage i-esimo e contiene coppie formate da una parte finita e una autorizzazione K_i, i=1,2.
La coda DA_i, contiene coppie formate da domande e autorizzazioni a produrre nuove parti finite relative allo stage i-esimo, i=1,2.
La coda P_0 rappresenta il buffer di parti grezze e la coda D_3 contiene la domanda dei clienti. La coda I_i rappresenta l'input buffer dello stage i-esimo, i=1,2.
Quando il sistema è allo stato iniziale, prima che ogni domanda abbia raggiunto lo stage finale, il buffer P_0 contiene un numero iniziale di parti grezze, la coda PA_i contiene K_i parti finite dello stage i-esimo ad ognuna delle quali è abbinato un kanban. Tutte le altre code sono vuote. Il parametro K_i risulta l'unico controllo sul sistema in esame [18].
Il sistema, seguendo lo schema contenuto in [18], opera nel seguente modo
Non appena giunge la domanda di un cliente, viene assegnata alla coda D_3. Parte così la richiesta di rilascio di una parte finita alla coda PA_2. Se in PA_2 vi è una parte disponibile, viene immediatamente rilasciata e contemporaneamente si provvede alla liberazione del kanban che era accoppiato alla stessa.
Questo kanban appena staccato viene trasferito a monte alla coda DA_2, portando con se l'informazione relativa alla domanda di parti finite della stazione 2 e autorizzando il rilascio di parti finite da PA_1 nella coda I_2. Se il numero di parti contenute nella coda PA_1 è sufficiente a soddisfare la domanda costituita dalle autorizzazioni K_2 presenti in DA_2, le parti richieste passano da PA_1 a I_2, dopo essersi liberate dei kanban K_1 e avere ricevuto i kanban K_2 che erano nella coda DA_2. I kanban K_1 liberati vengono inviati alla coda DA_1, portando con loro l'informazione relativa alla domanda della stazione uno e autorizzando il rilascio di parti grezze da P_0 a I_1.
In sintesi, il kanban che era associato alle parti che da PA_1 passano a I_2 viene inviato alla coda DA_1 come autorizzazione per la richiesta di materie prime. I kanban in DA_2 vengo associati alle parti in I_2 non appena i kanban sopracitati sono inviati alla coda DA_1.

La domanda viene trasferita dalla stazione 2 alla 1, tramite i Kanban K_2 che vanno da PA_2 a DA_2. Il trasferimento della domanda dalla stazione 1 alla stazione delle materie prime passa attraverso i kanban K_1 che vanno da PA_1 a DA_1. Nel caso in cui i pezzi alle code PA_i non fossero disponibili, allora il sistema si blocca perché si interrompe il flusso informativo e quindi la comunicazione tra gli stadi [18].
La filosofia dei sistemi KCS è la seguente: la domanda viene trasmessa a monte dallo stage i-esimo solo quando le parti relative a questo stage sono inviate verso valle verso lo stage i+1-esimo [18].
Il cartellino kanban adempie, in sintesi, a tre funzioni principali:
- consente di visualizzare il flusso informativo all'interno del sistema che corrisponde al flusso dei materiali, poiché ogni parte in lavorazione (WIP) può muoversi solo se ha attaccato il proprio cartellino kanban;
- ogni cartellino staccato in uno stadio a valle, svolge una funzione di controllo della produzione, indicando quantità e tipo di parte che deve essere prodotta a monte;
- il numero di kanban misura l'attuale livello di scorte: quindi controllare il numero di kanban corrisponde a controllare il livello di scorte, cosicché aumentare o diminuire il numero di kanban equivale ad aumentare o diminuire la quantità di materiale a scorta.

2.3.2.2 Modello Base Stock Control System - BSCS

Il sistema Base Stock, introdotto in [4] e [26], si basa sul mantenimento di un livello di scorta di sicurezza attraverso l'utilizzo di un buffer di stoccaggio tra le fasi di coordinamento della produzione.
Come per il KCS, per illustrare e analizzare il comportamento dei sistemi BSCS, si fa riferimento alla schematizzazione sviluppata da Liberopoulos e Dallery in [18].
La fig. 2.5 mostra il comportamento di un sistema BSCS composto da due stage in serie [18].
La coda P_i rappresenta l'output buffer relativo allo stage i-esimo, i=1,2. La coda D_i contiene la domanda per la produzione di nuove parti relativa allo stage i, i=1,2. La coda I_i rappresenta il buffer in ingresso dello stadio della produzione i-esimo (MF_i), i=1,2. La coda P_0 rappresenta la coda delle materie prime [18].

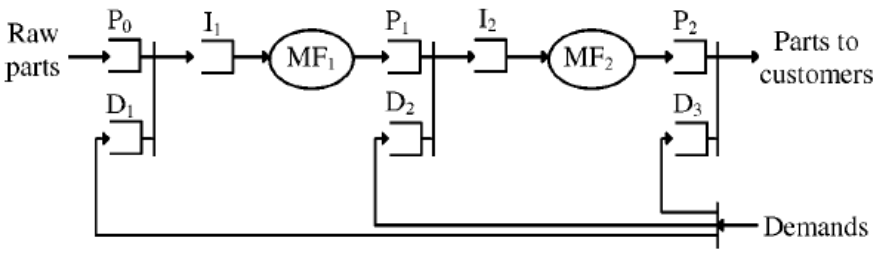

Figura 2.5. Sistema BSCS a due stage in serie

La coda D_3 infine rappresenta, la domanda di prodotti finiti.
Il sistema allo stato iniziale, prima dell'arrivo di ogni domanda del cliente, presenta nella coda P_0 un certo numero di parti grezze mentre la coda P_i contiene S_i parti finite per ogni stage della produzione i-esimo, i=1,2. Tutte le altre code sono vuote.
S_i è l'unico parametro di controllo presente in BSCS e rappresenta il livello di stock di ognuno degli stage del sistema. Nei sistemi BSCS non ci sono quindi autorizzazioni; c'è solo un livello di stock da mantenere.
Quando arriva la domanda del cliente alla coda D_3, essa viene istantaneamente inoltrata a tutti gli stadi attraverso le code D_i, i=1,2, autorizzando il rilascio delle parti da P_{i-1} a I_i, i=1,2. In sintesi, il trasferimento della domanda dalla coda D_3 alle code D_2 e D_1, consente il rilascio di parti, rispettivamente dalla coda P_1 e P_0 alla code I_1 e I_2.
La filosofia è la seguente: quando si concretizza la domanda di parti finite, il sistema la inoltra a tutti gli stadi, autorizzando la lavorazione immediata di parti, ammesso che in ogni buffer di monte il sistema abbia parti da inviare a valle [18].
Di conseguenza viene continuamente ricostituito il livello di stock che viene selezionato a priori .
Un vantaggio di questo meccanismo è che evita il blocco delle informazioni della domanda, trasferendola immediatamente a tutte le fasi di produzione (flusso informativo globale).
Tuttavia non viene controllato il numero di pezzi che circolano nel sistema, poiché ogni richiesta che arriva ad esso autorizza il rilascio di nuove materie prime nella prima fase. Un modo per superare questo inconveniente sarà quello di imporre un ulteriore meccanismo di controllo del WIP per ogni fase.
Se non vi sono prodotti finiti in magazzino quando arriva la domanda, essa è in backorder.La politica Base Stock cerca di mantenere una certa quantità di pezzi finiti in ciascun buffer di uscita. La quantità stabilita come scorta viene chiamata base stock level per ogni fase.
Il controllo Base Stock è un semplice meccanismo che dipende solo da un parametro per fase, cioè il livello di scorta S_i (base stock level) che influenza la velocità con cui la domanda viene soddisfatta. La capacità produttiva del sistema non dipende da S_i, ma è data dalla capacità di produzione della fase collo di bottiglia.

2.3.2.3 CONWIP Control System

Il sistema di controllo CONWIP fu proposto da Spearman in [27] e Framinan in [28] ed è un particolare esempio di KCS. L'idea è quella di controllare e mantenere costante il livello di WIP di tutto il sistema produttivo. Essa differisce dall'idea del KCS di controllare il livello di WIP di ogni stage. In altre parole, il

meccanismo di controllo del WIP localizzato che caratterizza il KCS, nei sistemi CONWIP, viene esteso a tutta la linea di produzione.
Quindi una volta raggiunto il numero di pezzi prefissato (livello di WIP limite) ogni volta che un pezzo esce per soddisfare la domanda di un nuovo cliente, ne entra uno grezzo alla prima stazione.
Questo sistema è un sistema che viene anche chiamato Alternative Kanban Control System.

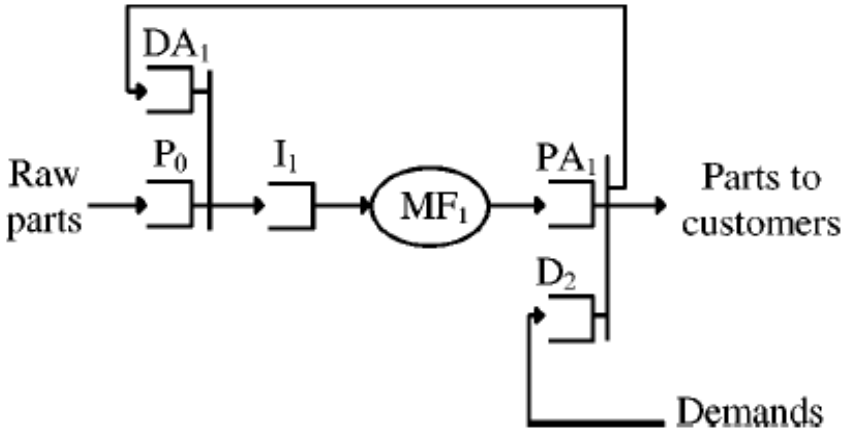

Figura 2.6. Sistema Conwip ad una stazione

La fig. 2.6 mostra il comportamento di un sistema CONWIP secondo lo schema proposto da Liberopoulos e Dallery in [18].
Il sistema CONWIP presenta vincoli per la prima e la ultima stazione. In un sistema composto da N stazioni di indice i-esimo, all'ultima stazione ci sarà la sincronizzazione tra la domanda e il buffer delle parti finite, rispettivamente D_{N+1} e PA_i, i=1,..,N. Alla prima stazione invece, la sincronizzazione, avverrà tra le parti contenute nel buffer delle materie prime P_0 e le richieste di parti alla prima stazione contenute nella coda DA_1. Il meccanismo legato alla richiesta di ingresso di nuove parti è controllato dalla sincronizzazione tra le code PA_i e DA_1, infatti, i kanban autorizzazione percorrono un tragitto che va dall'ultima alla prima stazione [18]. Il sistema CONWIP viene anche definito KCS a singola stazione. Nello schema, il sistema produttivo, è rappresentato una linea con una sola stazione. In questo tipo di logica di controllo della produzione, non interessa cosa c'è nella parte centrale di tale sistema. Infatti, qualunque sia il numero delle stazioni da cui è composta la linea, il comportamento del sistema CONWIP sarebbe sempre identico, infatti, il flusso informativo relativo alla domanda e il tragitto del kanban sarebbe sempre dall'ultimo stadio al primo.

2.3.2.4 Generalized Kanban Control System – GKCS

Il sistema di controllo ibrido Generalized Kanban Control System, che include il Kanban e il Base Stock, è stato proposto da Buzacott in [29] e Zipkin in [30].

Per ulteriori dettagli sul GKCS è possibile far riferimento alle trattazioni in [18], [31] e [32]. Il GKCS è un sistema molto versatile ma più rispetto ai controlli Kanban e Base Stock.

La complessità è dovuta principalmente dalla necessità di definire e gestire due parametri per stadio: la quantità di base stock e il numero di kanban. Di Mascolo, Frein e Dallery in [33], hanno studiato l'influenza di questi parametri sull'efficienza dei sistemi che adottano il GKCS. L'esempio seguente di GKCS, come per i modelli Pull precedenti, fa riferimento alla trattazione di Liberopoulos e Dallery in [18], fig. 2.7.

Il sistema in esame presenta due parametri per il controllo, K_i e S_i, rispettivamente il livello di kanban e di stock per ogni stage del sistema.

Il sistema che viene analizzato in questo lavoro ha $K_i > S_i$ [18].

La coda PA_i rappresenta l'output buffer dello stage i-esimo, i=1,2 e contiene coppie formate da parti finite e autorizzazioni alla produzione dello stage i-esimo, i=1,2. DA_i contiene coppie di autorizzazioni e domande per la produzione di nuove parti i-esime, i=1,2. La coda A_i contiene i "free" kanban dello stage i-esimo i=1,2. D_i contiene le domande per la produzione di nuove parti dello stage i-esimo, i=1,2.

P_0 rappresenta il buffer di prodotti grezzi e D_3 contiene la domanda di parti finite. La coda I_i rappresenta l'input buffer dello stage i-esimo, i=1,2.

Quando il sistema è allo stato iniziale, prima dell'arrivo di ogni domanda, la coda P_0 contiene un certo numero di parti grezze. La coda PA_i contiene un certo numero S_i di parti a stock a ognuna delle quali è accoppiato un cartellino kanban. La coda A_i contiene $K_i - S_i$ free kanban relativi allo stage i-esimo, i=1,2, mentre tutte le altre code sono vuote.

Nel GKCS, inizialmente, ci sono S_i kanban attaccati ad un uguale numero di parti finite all'interno della coda PA_i, ma ci sono anche $K_i - S_i$ free kanban nella coda A_i. Questi kanban extra, forniscono un parziale disaccoppiamento tra il trasferimento di parti a valle dello stage i-esimo e il trasferimento a monte della domanda presente in DA_i. Se per esempio K_i fosse uguale a S_i, per tutti gli stage i-esimi, il sistema GKCS sarebbe identico al KSC, vista l'assenza dei cartellini free kanban.

L'utilizzo di due parametri per ogni stage i-esimo, K_i ed S_i, e la presenza dei free kanban, rendono possibile quindi il trasferimento della domanda di parti anche se non è ancora avvenuto l'effettivo completamento di una parte. Questo significa che i sistemi GKCS rispondono alla domanda più velocemente rispetto ai sistemi KCS. I tempi di risposta migliorano all'aumentare del livello di kanban free presenti nel sistema.

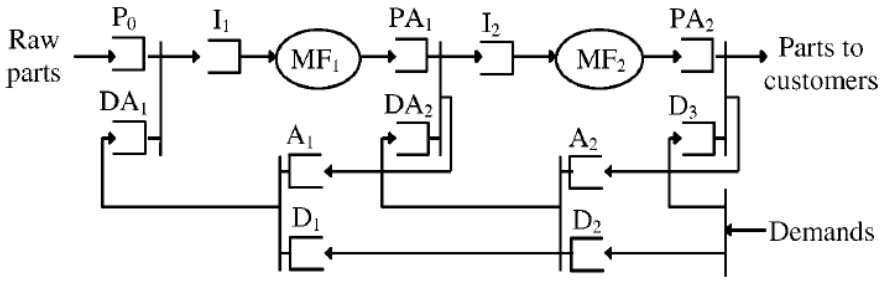

Figura 2.7. Sistema GKCS con due stage in serie

Il GKCS comprende entrambe le politiche di base da cui è composto come casi particolari:
• quando $K_i = S_i$ per ogni fase i, allora il GKCS coincide con il controllo Kanban tradizionale di parametri K_i per ogni fase i;
• nel caso in cui $K_i = \infty$ e $S_i \geq 0$ per ogni fase i, allora il GKCS è equivalente al BSCS con lo stesso base stock level S_i per ogni fase i.
La capacità produttiva del GKCS dipende da entrambi i parametri di controllo.

2.3.2.5 Extended Kanban Control System – EKCS

L'Extended Kanban Control System è stato introdotto da Dallery e Liberopoulos in [18] e rappresenta un'alternativa ai sistemi Kanban, Base Stock e Generalized Kanban, nessuno dei quali raggiunge un giusto equilibrio tra la risposta rapida ai clienti e un basso valore di Work-In-Process.
Il modello EKCS, in fig.2.8, fa riferimento alla schematizzazione proposta da Dallery e Liberopoulos in [18].

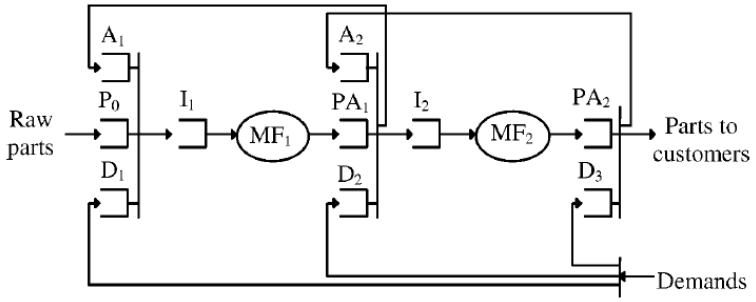

Figura 2.8. Sistema EKCS a due stage in serie

Il sistema è composto da N=2 stazioni i-esime, monoprodotto. La coda PA_i rappresenta l'output buffer dello stage i-esimo e contiene coppie di parti finite e autorizzazioni, relative allo stage i-esimo, i=1,2. La coda A_i contiene i free kanban relativi allo stage i-esimo, i=1,2. La coda D_i contiene la domanda per la produzione di nuove parti finite allo stage i-esimo, i=1,2 mentre la coda P_0 rappresenta il buffer delle parti grezze. La coda D_3 contiene la domanda dei clienti e la coda I_i rappresenta l'input buffer dello stage i-esimo, i=1,2.

Allo stato iniziale, prima dell'arrivo della domanda, il sistema parte nella medesima situazione descritta per il GKCS ossia, con un certo numero di parti grezze in P_0, un certo livello S_i di parti finite nella coda PA_i e un certo numero di free kanban contenuti nelle code A_i, i=1,2. Tutte le altre code sono vuote.
La dinamica del sistema EKCS è la combinazione delle dinamiche dei sistemi KCS e BSCS; non appena la domanda di parti finite giunge al sistema, viene comunicata a tutti gli stadi della produzione (BSCS) tuttavia, il trasferimento delle parti verso valle viene autorizzato solo in presenza di un determinato numero di autorizzazioni kanban (KCS), come affermato in [18].
La domanda di parti finite alla coda D_{N+1}, genera automaticamente una domanda alle code D_i, i=1,2. In questo sistema Pull, il flusso informativo della domanda non viaggia attraverso i kanban ma viene instradato a tutte le stazioni, all'arrivo della domanda di parti finite, come nei sistemi BSCS. L'unica differenza tra EKCS e BSCS è che l'EKCS, per cedere una parte finita presente nella coda PA_i, necessita di una autorizzazione kanban.
Questa autorizzazione kanban viene generalmente presa dalla coda A_i contenente i free kanban. La principale differenza tra EKCS e GKCS è che nell'EKCS il ruolo di K_i e Si risulta chiaramente distinguibile. Questo ci permette di dire che nell'EKCS la capacità produttiva dipende esclusivamente da K_i. Questo perché in condizione di sistema saturo, si comporta come un Kanban e il throughput di un KCS dipende da K_i.
I kanban si muovono a monte separatamente dalla domanda, mentre i pezzi vengono trasferiti sempre insieme ai propri kanban, che non vengono subito staccati dai pezzi alla fine di ogni lavorazione, come accade nel GKCS. L'EKCS, quindi, dipende da due parametri per fase, numero di kanban cards K_i e livello target di scorta S_i .
L'Extended Kanban Control System impone, però, un vincolo tra i due parametri: prevede che il numero di kanban cards per ogni fase sia maggiore del numero del livello di base stock stabilito.
In determinate condizioni, l'EKCS si riduce alle politiche di base Kanban e Base Stock. In particolare:

- l'EKCS è equivalente al KCS se risulta $K_i = Si$ per ogni fase i;
- se risulta $K_i = \infty$ e $S_i \geq 0$ per ogni fase i, l'EKCS è equivalente al BSCS con lo stesso base stock level S_i per ogni fase i.

La capacità produttiva di questa logica di controllo della produzione dipende solo dal numero di kanban K_i in ogni fase i.

2.3.3 Confronto tra le principali tecniche

In letteratura sono state presentate moltissime trattazioni relative all'analisi delle singole strategie di controllo della produzione pull, mentre è possibile individuare solo pochi studi relativi al confronto fra le varie tecniche; ciò è dovuto, in parte, al fatto che le diverse politiche di controllo sono state

analizzate ipotizzando contesti differenti che non hanno permesso uno studio comparativo.
In genere gli studi condotti dai vari autori mostrano che le prestazioni dei diversi sistemi possono variare anche significativamente, a parità di condizioni ipotizzate.
Il meccanismo del controllo del WIP rappresenta uno degli aspetti alla base del confronto di alcune tecniche. In particolare,con il termine controllo del WIP si fa riferimento a due casi, il controllo locale e quello globale: nel primo caso si limita, separatamente, la massima quantità di work in process in ogni fase del sistema, nel secondo caso si limita la massima quantità totale di WIP nel sistema.
Se si confrontano i sistemi di controllo in funzione del WIP, mentre il Kanban esegue un controllo locale il Conwip consiste in un controllo globale.
Un secondo aspetto rilevante in termini di confronto tra tecniche è rappresentato dal trasferimento del flusso informativo. In particolare, questo può essere classificato come accoppiato, parzialmente accoppiato o disaccoppiato al meccanismo di controllo del WIP, a seconda della modalità con cui le informazioni relative alla domanda vengono trasmesse nel sistema.
Nel caso di sistema Kanban, le informazioni della domanda vengono trasferite a monte solo quando le parti vengono prelevate da valle: in questo caso, il flusso informativo è completamente accoppiato al controllo del WIP.
Questa condizione viene rilassata nei sistemi CONWIP e GKCS. Nel primo l’informazione è trasferita alla prima fase del sistema,ma solo quando il buffer prodotti finiti risulta non vuoto; nel GKCS il trasferimento della domanda a monte non è completamente sincronizzato con il trasferimento del materiale a valle. Per questo sistema il flusso informativo risulta, infatti, parzialmente accoppiato al meccanismo di controllo del WIP.
Per tutte le altre politiche l’informazione relativa alla domanda viene trasferita direttamente a tutte le fasi non appena la domanda giunge al sistema: il trasferimento delle informazioni è completamente disaccoppiato al controllo del WIP.
Il disaccoppiamento tra flusso informativo e movimentazione dei materiali deriva dalla necessità di stabilire la quantità opportuna di materiale da mantenere nel sistema per raggiungere il giusto trade-off tra rapidità di risposta alle variazioni di domanda e bassi costi legati al WIP. Infatti, elevati livelli di WIP e prodotti finiti nel sistema garantiscono di rispondere rapidamente all’aumento della domanda, ma determinano contemporaneamente alti costi di mantenimento.
Questo conflitto è evidente nel meccanismo Kanban, in cui l’unico parametro di controllo per fase, ovvero il numero di kanban, gioca due ruoli opposti:in primo luogo serve a limitare il livello di WIP in ogni fase (minore è il numero di cartellini, minore è la quantità di WIP) e in secondo luogo, fornisce il livello target nell’output buffer di ogni fase necessario a fronteggiare eventuali

interruzioni e variazioni nel sistema (necessità di un maggior numero di cartellini).
Questa caratteristica "due ruoli in un parametro" può portare a cattive prestazioni del sistema soprattutto quando la domanda effettiva o tempi di lavorazione sono molto variabili. Per esempio, in una situazione di variabilità elevata della domanda, si vorrebbe avere un numero elevato di Kanban, per rispondere rapidamente alla domanda. Allo stesso tempo, si vorrebbe avere un numero ridotto di kanban in periodi di scarsa domanda, per ridurre i costi di magazzino, poiché il numero di kanban è uguale al livello target del buffer di pezzi finiti. Quindi il sistema non può svolgere al meglio i due ruoli in condizioni di forte variabilità di domanda o processo.
Lo stesso problema si verifica implementando il controllo Base Stock: un elevato livello di base stock da un lato permette di compensare gli aumenti nelle richieste, ma dall'altro determina un maggiore livello di WIP medio nel sistema.
Queste criticità hanno portato alla necessità di politiche caratterizzate da più parametri di controllo sebbene più complesse e, quindi, di ardua implementazione. Questo disaccoppiamento avviene nel GKCS, considerando due parametri differenti per fase, numero di kanban e base stock level. Essi consentono di trasferire la domanda a monte senza la necessità di avere parti finite nell'output buffer. Il disaccoppiamento, in questo caso, è totale. Nell'EKCS, invece, la distinzione completa degli ruoli dei due parametri di controllo consente di trasferire direttamente la domanda a tutte le fasi in seguito all'arrivo della stessa nel sistema.
L'implementazione di EKCS è concettualmente più semplice rispetto a GKCS perché il flusso informativo viaggia contemporaneamente in tutte le stazioni ogni volta che allo stadio di domanda giunga una richiesta di parti finite. Nei sistemi EKCS la capacità produttiva dipende solo dal livello di K_i.
Nel modello GKCS, la fase di propagazione verso monte della domanda si comporta come nei sistemi KCS; in questo caso però, arretra anche se non vi sono parti finite nell'output buffer, in seguito alla presenza dei free kanban.
In questi sistemi GKCS la capacità produttiva dipende da entrambi i parametri di controllo K_i ed S_i.

2.4 Tecniche di Simulazione e Ottimizzazione per politiche Pull

In questa sezione vengono presentate le tecniche di simulazione e ottimizzazione utilizzate per analizzare e ottimizzare i sistemi governati dalle politiche di controllo della produzione di tipo Pull.
Dalla simulazione è possibile ottenere gli indicatori delle prestazioni del sistema produttivo. In fase di ottimizzazione, invece, si ricavano i valori ottimi dei parametri di controllo della politica adottata.

In questa sezione verranno sviluppati i modelli di simulazione e ottimizzazione che riguardano le tecniche Kanban Control System e Base Stock Control System.
Le principali tecniche di simulazione sono: simulazione ad Eventi Discreti (DES), metodi analitici basati su reti di code e programmazione matematica.
Quest'ultima verrà trattata separatamente nella sezione 2.5 essendo l'approccio adottato nel presente lavoro.
Per quanto riguarda le tecniche di ottimizzazione, in letteratura si trovano numerose trattazioni che riguardano: tecniche Euristiche, Programmazione Matematica, tecniche meta-euristiche, modelli di ottimizzazione iterativi basati su simulazione (Response Surface Methodology).

2.4.1 Sistemi KCS

Per quanto riguarda gli approcci simulativi, si ricordano i metodi Analitici e di simulazione ad Eventi Discreti (DES).
Per la fase di ottimizzazione sono stati sviluppati metodi Euristici e modelli di Programmazione Matematica.

2.4.1.1 Tecniche di Simulazione per KCS

In letteratura sono stati proposti numerosi modelli analitici per la valutazione delle performance dei sistemi KCS. Di Mascolo in [34] propone un'analisi basata su modelli di reti di code. L'algoritmo proposto, tuttavia, risulta particolarmente oneroso dal punto di vista del tempo impiegato per la risoluzione.
Baynat, Di Mascolo, Frein, Dallery in [35] hanno proposto un algoritmo per un sistema KCS multi stage basato sulla tecnica product-form-approximation, che si è dimostrato più efficiente rispetto a quello dello studio di partenza sviluppato da Di Mascolo in [36]. Questo approccio consente, inoltre, di ottenere gli stessi indicatori delle performance ricavati in [36]. Un altro vantaggio è che permette l'estensione del modello anche a sistemi con venditori multipli e clienti multipli.
Un ulteriore Metodo Analitico Approssimato per la misura delle performance nei sistemi KCS, è presentato da Di Mascolo, Duri, Frein in [37].
Questo metodo permette di studiare il sistema dove le code di ogni stage della produzione hanno una capacità più ampia rispetto al numero di kanban presenti nello stage stesso, escludendo così il fenomeno di blocking. Il modello di risoluzione basa sugli studi della product-form-approximation; l'idea di base è quella di approssimare la misura delle performance utilizzando un set di reti di code chiuse approssimate (closed single class product-form queueing networks). Questo modello consente di ottenere indicatori delle prestazioni come il livello di WIP o il numero medio di parti finite per stage o il numero medio di domande in attesa o la percentuale di domande in backorder. Una successiva analisi numerica, ha evidenziato sia l'attendibilità dei risultati che la velocità di ottenimento degli stessi.

Una seconda classe di metodi, utilizza modelli stocastici a tempo continuo. Il comportamento del sistema è quindi rappresentato da una catena di Markov a tempo continuo (CTMC). Karmarkar e Kekre in [38] analizzano sistemi kanban ad uno stadio e a due stadi, risolvendo catene di Markov a tempo continuo associate a tecniche numeriche.
Soluzioni analitiche esistono quasi esclusivamente per sistemi seriali con tempi distribuiti deterministicamente o esponenzialmente (i riferimenti sono contenuti in [29], [39], [40] e [41]) .
La maggior parte dei metodi analitici che sono stati proposti per i sistemi kanban assumono che ogni stadio consista in un singola macchina; ciò significa che valgono per sistemi kanban semplici.
Recentemente alcuni metodi di analisi sono stati sviluppati su modelli a tempo discreto. Kimura e Terada in [42] forniscono le equazioni di base per una sistema kanban composto da fasi in serie. Questi due modelli sono deterministici, e quindi non incorporano incertezze.
I modelli matematici utilizzati includono, oltre ai modelli analitici (processi di Markov, reti di code), anche modelli di simulazione.
Deleersnyder in [39] utilizza una catena di Markov a tempo discreto di un modello di sistema kanban con stadi in serie. Questo modello è stocastico perché tiene conto dei guasti delle macchine, utilizzando tempi di servizio random, e una domanda casuale.
L'uso di reti di Petri per la modellazione dei sistemi kanban è stato suggerito da Di Mascolo in [36] e da Legato, Bobbio e Roberti in [43].
Tra le metodologie di simulazione più utilizzate negli ultimi anni, troviamo l'utilizzo di software dedicati. Il software, una volta implementato il modello di simulazione, è in grado di ottenere una grande varietà di indicatori delle prestazioni del sistema, senza richiedere assunzioni forti come nei modelli analitici.

2.4.1.2 Tecniche di Ottimizzazione per KCS

Negli ultimi decenni sono stati presentati diversi modelli per l'ottimizzazione del numero di kanban. Philipoom, Rees e Taylornel in [44], hanno utilizzato un approccio di programmazione intera per analizzare differenti sistemi a Kanban considerando i tempi di setup delle macchine.
Al-Tahat nel riferimento [45] sviluppa un modello a stage singoli sincronizzati per linee di produzione KCS. La ricerca fu ampliata da Al-Tahat stesso e da Mukattash in [46]. Nel loro modello hanno considerato quattro componenti di costo diverse: il costo dei materiali grezzi, il costo manifatturiero, il costo di trasporto e il costo di mantenimento a scorta. Questa funzione di costo complessivo è stata minimizzata, ottenendo il livello di Kanban ottimo.
Partendo dallo studio sulle politiche di consegna in ambiente JIT, nascono gli studi su sistemi di produzione multi stage KCS.

Wang e Sarker in [47] estesero gli studi di Sarker e Balan in [2], introducendo il costo di kanban inteso come la somma del costo di produzione e costo di trasporto. Rabbiani, Layegh e Ebrahim in [48] estesero il lavoro di Wang e Sarker in [47], introducendo il costo complessivo come la somma di costo di trasporto, di setup e di mantenimento a scorta, sviluppando una procedura per la risoluzione di un sistema multi-stage monoprodotto. Chen, Guo e Lim in [49] svilupparono un modello matematico per minimizzare il costo totale di trasporto e di mantenimento a scorta in funzione del livello di kanban, considerando anche i tempi di trasferimento delle parti e la capacità dei magazzini.
Per aumentare l'efficienza e ridurre lo sforzo computazionale richiesto da questi metodi di ottimizzazione, nel corso degli anni sono stati sviluppati una serie di modelli euristici.
Mooeni e Chang in [50], svilupparono un modello euristico per determinare il numero ottimo di kanban in un sistema di produzione multi stage che lavora un singolo prodotto alla volta. Bart e Golany nella pubblicazione in [51], determinarono il numero ottimo di kanban in un sistema multi prodotto multi stage utilizzando un modello euristico generale. Chan, Yih in [52] svilupparono il sistema introdotto in precedenza da Chan e Yih utilizzando la metodologia Tabu Search e Simulated Annealing, scoprendo che il loro modello, rispetto ai precedenti, otteneva risultati più vicini alla soluzione ottima. Shahabudeen, Gopinath e Krishnaiah [53], determinarono il numero ottimo di kanban utilizzando proprio un modello con funzione di costo minimizzata attraverso la fase di Simulated Annealing.
Al Tahat [46] sviluppò un algoritmo genetico (GA) per un sistema di produzione che adotta la politica Kanban Control System. Questo modello meta-euristico è stato ulteriormente sviluppato da Chen in [49] per ottimizzare sistemi JIT anche in condizione di cross-docking (operazione per cui la merce che arriva da più destinazioni viene scaricata e, almeno in parte, ricaricata direttamente su altri mezzi, senza sosta a terra). La novità dello studio di Chen [49] è dettata dal fatto che è stata integrata la logica Tabu Search in una fase Simulated Annealing in un sistema cross-docking.
Kochel e Nielander in [54] hanno proposto l'uso di algoritmi genetici (GA) e la simulazione ad eventi discreti per ottimizzare il numero di kanban per le linee di produzione con tempi di trasferimento random e macchine affidabili.
Widyadana, Wee e Yuan Chang ne lavoro in [55] hanno introdotto un modello per sistemi multi stage multi prodotto risolto utilizzando una mix tra programmazione lineare intera, algoritmi genetici (GA) e Hybrid Genetic Algorithm and Simulated Annealing (GASA).
Un ulteriore modello di ottimizzazione è presentato da Di Mascolo Duri Frein in [37]. Questo modello ha come dati di input, gli indicatori delle performance ottenuti con il modello di simulazione analitico approssimato. Alla base di questo modello di ottimizzazione, vi è la creazione di una funzione di costo complessivo e la definizione del concetto di probabilità di backlog. La funzione

di costo è data dalla somma del costo associato al livello di WIP e dal costo associato al numero medio di parti finite.
Per quanto riguarda la definizione di probabilità di backlog, si intende la probabilità che caratterizza una parte che giunge in ritardo, di avere davanti a se un certo numero di parti già in backlog. In questo modello, il livello di kanban influenza sia il valore della funzione di costo che l'andamento della probabilità di backlog. L'ottimizzazione avviene attraverso un algoritmo iterativo che, facendo variare la configurazione del livello di kanban, trova il punto di minimo della funzione di costo complessivo. I risultati sono forniti in funzione della probabilità di backlog.
Murino, Naviglio, Romano, Zoppoli in [56] trattano di sistemi di ottimizzazione e analisi parametrica del livello di kanban in un sistema single stage multi prodotto utilizzando come ottimizzatore, un tool (OptQuest for Arena®) inserito all'interno di un software di simulazione dedicato (Arena - Rockwell®). Essi hanno sviluppato una funzione di costo complessiva ottenuta dalla somma del costo di backlog e del costo di mantenimento a scorta. Grazie all'ausilio del software di ottimizzazione, ottengono il livello di Kanban che minimizza la funzione di costo.

2.4.2 Sistemi BSCS

Per quanto riguarda gli approcci simulativi, si ricordano i Modelli Analitici e i Simulatori ad Eventi Discreti (DES).
Per la fase di ottimizzazione sono stati sviluppate tecniche Euristiche, Programmazione Matematica, tecniche meta-euristiche, modelli di ottimizzazione iterativi basati su simulazione (Response Surface Methodology).

2.4.2.1 Tecniche di Simulazione per BSCS

In letteratura si trovano numerosi modelli analitici per l'analisi delle prestazioni di sistemi BSCS.
Le trattazioni più importanti sono quelle di Buzacott [57] e di Svoronos e Zipkin [58]. Il modello di Svoronos e Zipkin [58], considera un sistema multi stage dove ogni ognuno di essi è frapposto tra due stazioni di sincronizzazione.
All'interno del modello vengono definiti: il numero delle parti finite in ogni stazione di sincronizzazione, il numero di domande in attesa in una stazione di sincronizzazione i-esima e il massimo numero di parti finite che possono essere allocate su ognuna delle stazioni di sincronizzazione. A partire da questi indicatori, sono definiti il tempo di attesa della domanda su una i-esima stazione di sincronizzazione e il valore del flowtime (tempo di attraversamento) di un i-esimo processo manifatturiero.
Questo metodo è basato sulle proprietà delle distribuzioni phase-type introdotte da Neuts in [59]. Partendo dai dati analitici sopracitati, attraverso le distribuzioni di Neuts, sono state ricavate alcune particolari distribuzioni di probabilità dalle quali è possibile desumere le performance del sistema produttivo.

Lee e Zipkin in [60] hanno applicato il modello a sistemi multi stage con tempi di lavoro esponenziali schematizzando opportunamente la linea come una serie di single server con tempi di servizio di tipo esponenziale e di tempi di interarrivo della domanda poissoniani. Grazie a questa semplificazione, ogni server è stato gestito come una coda M/M/1, ottenendo risultati in modo più rapido
Frein, Duri, Di Mascolo [61] hanno esteso questa analisi a sistemi multi stage sviluppando un modello che consentisse di utilizzare distribuzioni dei tempi di servizio anche più complesse rispetto all'esponenziale.
Come detto in precedenza per i modelli KCS, anche nei sistemi BSCS, l'analisi fatta mediante simulazione ad eventi discreti (DES), viene utilizzata sia come modello per la determinazione delle performance dei sistemi produttivi, che come riscontro per la validazione dei risultati ottenuti dai modelli analitici o analitici approssimati.
Anche nei sistemi BSCS, la fase di simulazione può essere eseguita con l'ausilio di software dedicati. La modellizzazione del sistema produttivo BSCS, eseguita in ambiente software, permette al programma di simulare le dinamiche del sistema produttivo e di definire gli indicatori delle performance di cui l'utente necessita, in tempi brevissimi.

2.4.2.2 Tecniche di Ottimizzazione per BSCS

Tra le prime trattazioni in materia di ottimizzazione del livello di inventory o livello di base stock, si può trovare la pubblicazione di Clark e Scaarf in [4].
In questo modello, viene sviluppato un sistema produttivo costituito da una serie di stage multipli. I due, hanno preso in considerazione i tempi discreti, un orizzonte temporale finito ed una domanda distribuita esponenzialmente utilizzando costi di mantenimento a scorta e di backlog lineari. Partendo da questi due costi, in contrapposizione, hanno determinato una funzione di costo totale di forma convessa, dimostrando che il livello ottimo di base stock è ottenuto dalla minimizzazione della funzione di costo totale. Il modello di Clark e Scaarf, se implementato sui sistemi sui quali è stato studiato, fornisce quindi risultati esatti.
Il problema di questo modello è che, risolvendo contemporaneamente una serie di problemi di ottimizzazione di funzioni, necessita di una grande potenza di calcolo e può subire rallentamenti durante la risoluzione. Federgruen e Zipkin [5] hanno semplificato l'algoritmo del modello di Clarke Scaarf, ottenendo una soluzione ottima anche per un orizzonte infinito. Rosling in [6], Langenhoff e Zijm [7] fornirono ulteriori studi sul modello di Zipkin e contribuirono all'analisi dei risultati. Chen e Zheng [8] estesero i risultati delle trattazioni precedenti e ottennero una soluzione esatta anche in condizioni di continuità introducendo un modello di ottimizzazione a orizzonte infinito sviluppato in dominio di continuità. Questo modello si affida all'ottimizzazione di una funzione di costo totale ottenuta dalla somma del costo di mantenimento a scorta

e del costo di backorder e minimizzata in corrispondenza del livello ottimo di base stock.
La complessità dei sistemi produttivi attuali e il grande sforzo computazionale necessario per la risoluzione del modello di ottimizzazione esatto, hanno fatto si che nel corso degli anni si sviluppassero modelli euristici per il calcolo del livello ottimo di Base Stock. Questi modelli, forniscono risultati vicini alla soluzione ottima, utilizzando tempistiche e risorse molto più limitate.
Zijm e Van Houtum nella pubblicazione in [9] hanno introdotto un modello euristico utilizzabile in sistemi multi stage che evidenziata la differenza tra livello di stock locale e di stock echelon.
Per livello di stock echelon su una data stazione, si intendono tutte le installazioni di inventory presenti in uno stage a cui si aggiungono le parti in transito o a disposizione meno eventuali backlog verso le stazioni di valle.
Il livello di stock locale, invece, è rappresentato dal livello di inventory presente su quella stazione. In questo modello viene presentata una funzione di costo totale, che dipende dal costo di mantenimento a scorta e dal costo di backlog. Attraverso la minimizzazione di queste funzioni di costo, viene calcolato il livello di base stock che caratterizza ogni stage produttivo.
La novità introdotta da questo studio, è rappresentata dal confronto tra i risultati del modello euristico e i risultati ottenuti con il Material Requirement Planning.
Da questo confronto si può notare la bontà dei risultati dell'euristico che denotano un miglioramento del 10% dei costi di gestione relativi allo stock rispetto i costi sostenuti dal modello MRP.
Nella pubblicazione in [10], Gallego e Zipkin presentano un metodo euristico per l'ottimizzazione del livello di inventory in sistemi multi stage. Questo modello si basa sulla definizione di una funzione di costo caratterizzata dalla somma del costo digestione e mantenimento della scorta e del costo di backlog. In questa pubblicazione Gallego e Zipkin sviluppano tre modelli euristici.
Il primo metodo euristico, chiamato Restriction e Decomposition Approximation Heuristic (RD), minimizza la funzione di costo, al variare del livello di base stock, utilizzando la stessa procedura adottata dall'algoritmo di Wagner Within nella dinamica del lot-sizing.
Il secondo modello euristico presentato, dal nome di Zero Safety Stock Heuristic (ZS), viene risolto mediante la scomposizione della linea produttiva in una serie di single stage. Per ogni stazione che caratterizza la linea di produzione, viene determinato il valore del livello di base stock che minimizza il costo di mantenimento a scorta e il costo di penalità da backlog.
Fondamentalmente viene eseguita una minimizzazione in locale, cioè su una stazione alla volta.
Il terzo modello euristico presentato prende il nome di Two Stage Heuristic (TS). Questo approccio, che utilizza lo stesso algoritmo del modello RD, restringe il calcolo del livello ottimo di inventory a due stage alla volta. Quindi

l'algoritmo iterativo, non viene più applicato a tutti gli stage contemporaneamente, ma sottosistemi composti da una coppia di stage.
Viene preso in considerazione una coppia di stage alla volta, andando a determinare il valore ottimo di base stock per ognuno di essi. Se il sistema fosse composto da J stazioni, questo euristico risolverebbe J-1 problemi di ottimizzazione a due stage.
I risultati degli algoritmi euristici sono influenzati dal tipo di domanda che caratterizza il sistema. Tuttavia, la differenza tra i risultati di questi euristici e quelli del modello di ottimizzazione esatta è minima. Per esempio, nel caso di domanda con una funzione a scalino (funzione che simula al meglio le variazioni improvvise del mercato) la differenza tra l'ottimo e i tre euristici è: per RD del 5-7%, per ZS 11-15% e per TS 1-3%.
Il tempo di calcolo di questi euristici, però, è notevolmente inferiore rispetto all'ottimizzazione esatta.
Shang e Song [11] introducono un modello euristico per l'ottimizzazione approssimata del livello di inventory. Questo approccio ha un contributo duplice: per prima cosa, fornisce un vantaggio computazionale e di implementazione mentre in secondo luogo, aggiunge trasparenza alla trattazione riguardante l'ottimizzazione del livello di inventory, che nel corso degli anni è sempre stata considerata come una fase chiusa e governata da "black box" euristiche. Il sistema si fonda sulla teoria del "newsvendor problem".
In questo modello viene definita una funzione di costo somma del costo di inventory (inteso come mantenimento a scorta) e del costo di backorder. Definita la funzione di costo, viene risolto un problema analogo al "newsvendor" ottenendo in questo modo, il valore del livello delle scorte ottimo. Da questa considerazione, l'euristico propone due funzioni costo per ogni stage che sono dipendenti da una serie di costi localizzati: dal costo di mantenimento a scorta, dal costo di backorder, dal tempo di durata, su una data stazione, del costo di mantenimento a scorta associato ad una i-esima parte e dal numero medio di costi di holding/backorder che si verificano su una stazione per il passaggio di una singola parte.
Per ogni stadio della produzione, si avranno due funzioni di costo definite opportunamente che limitano superiormente e inferiormente la funzione ottima stessa. In pratica, pur non essendo a conoscenza dell'andamento della funzione di costo ottima, le due funzioni di costo gestite dell'algoritmo euristico, riescono ad esserle molto vicino andando a limitarla superiormente e inferiormente.
Per ogni stage, l'algoritmo euristico permette di ottenere due funzioni limite che fungono da "binari", all'interno dei quali è presente la funzione ottima.
Ulteriori tecniche per l'ottimizzazione del livello di base stock, derivano modelli di ottimizzazione iterativi basati su simulazione (Response Surface Methodology). Tra i più utilizzati, si ricorda OptQuest for Arena®.

2.5 Mathematical Programming Representation (MPR)

In questa sezione viene presentata la modellizzazione mediante Programmazione Matematica (MPR) di sistemi di produzione Pull.
Questo approccio risulta di fondamentale importanza per la comprensione di questo lavoro di Tesi. Recentemente la programmazione matematica è stata utilizzata per rappresentare la dinamica dei sistemi a eventi discreti (DES). Questa trattazione è sviluppata negli studi di Chan e Schruben [12] e di Schruben in [62].
Un sistema DES può essere schematizzato attraverso un modello di programmazione matematica che fornisce lo stesso set di risultati della simulazione. In particolare, la dinamica del comportamento del sistema in esame è rappresentata con un modello di ottimizzazione nel quale la somma degli eventi che caratterizzano le attività di fine e inizio lavorazione è minimizzata.
I modelli MPR e DES forniscono gli stessi risultati perché nell' approccio MPR gli eventi vengono sequenziati nello stesso ordine di un sistema DES.
Questo sistema vincola il fluire delle parti all'interno del sistema e può modellizzare sistemi con buffer a capacità limitata (Chan e Schruben in [63]), numero di clienti costante e operazioni di assemblaggio (Matta, Chefson in [64]). Il modello restituisce le stessa sequenza di eventi della simulazione [12] e [62].
Il principale vantaggio di questo approccio è espresso dalla trasparenza e dalla semplicità dovuta controllo del modello, effettuato mediante l'utilizzo di equazioni lineari (funzione obbiettivo e vincoli). Questo porta degli ulteriori vantaggi sia per quanto riguarda la modifica e lo sviluppo del modello che per la comprensione e validazione dei risultati.
Il modello è composto da una funzione obbiettivo e da una serie di vincoli. Viene risolto mediante l'utilizzo del metodo del simplesso.

2.5.1 Modelli MPR per sistemi di tipo Pull

In questa sezione è proposta la trattazione per potere modellizzare correttamente un sistema produttivo Pull attraverso un modello di programmazione lineare MPR sviluppato da Matta e Alfieri in [13], [14] e [65].
Il fine è quello di ricavare un modello di programmazione lineare che sia in grado di simulare un sistema ad M-stages controllato con delle politiche di tipo pull.
Le politiche che si è deciso di prendere in considerazione in questo lavoro di Tesi sono: KANBAN e BASE STOCK.

2.5.1.1 Notazione utilizzata e assunzioni

Si comincia l'introdurre la notazione che verrà utilizzata in questo capitolo per la scrittura del modello matematico [14]. Si consideri un sistema con N parti da lavorare e M macchine per la produzione.

Ogni macchina costituisce uno stage e quindi si hanno a disposizione M stages in tutto il sistema produttivo.
Ogni i-esima parte prodotta va da 1 a N.
Ogni ordine da parte dei clienti viene ipotizzato costituito da una sola parte. Tuttavia si deve disaccoppiare il concetto di parte, cioè l'elemento prodotto dalla macchina, da quello di cliente, cioè la domanda che arriva all'ultimo stage del sistema. Quindi in condizioni di regime, la i-esima domanda che arriva all'ultimo stage, sarà soddisfatta dall'i-esima parte che viene prodotta dal sistema.
Il concetto di macchina, stazione, stage è il medesimo; così come quello di parte, item o prodotto.
Gli stage produttivi sono caratterizzati dall'indice j che va da 1 a M, mentre lo stage M+1 rappresenta lo stage di soddisfazione della domanda della clientela, cioè il pezzo domandato che giunge al cliente. L'arrivo delle parti e il loro processo produttivo sono governati da processi stocastici.
Nel modello di programmazione lineare vengono inizializzate tre variabili, dipendenti dal numero delle parti e dal numero degli stadi della produzione; esse sono : $X_{i,j}$, $Y_{i,j}$, $Z_{i,j}$.
$X_{i,j}$ rappresenta il minimo tempo di inizio processamento della parte i sulla macchina j, $Y_{i,j}$ rappresenta il minimo tempo di fine processamento della parte i sulla macchina j, $Z_{i,j}$ rappresenta il minimo tempo di rilascio della parte i sulla macchina j: la variabile Z è dovuta al fatto che un pezzo può essere rilasciato ad uno stage (e di conseguenza lavorato e processato nel minor tempo possibile) per poi passare allo stage successivo che, una volta disponibile, provvederà ad eseguire la lavorazione successiva [14].
Le parti non si scavalcano, la logica di processamento per tutto il sistema è FIFO, per cui l'ordine di arrivo rispecchia l'ordine di processamento sulle stazioni e l'ordine di uscita dal sistema.
Sono definiti anche i tempi di arrivo delle parti grezze, i tempi di lavorazione, i tempi di interarrivo della domanda e i valori relativi ai parametri kanban e base stock.
Con a_i si intende il tempo di arrivo nel sistema delle materie prime necessarie per il prodotto i- esimo. Con $t_{i,j}$ è rappresentato il tempo di processamento per l'i-esimo prodotto sulla j-esima macchina.
Con D_i si intende l'istante di arrivo della domanda dell'i-esimo cliente all'ultimo stage.
Con K_j si intende il numero di kanban/autorizzazioni presenti allo stage j mentre con S_j è espresso il livello di base stock al j-esimo stage.
Tutte le politiche di tipo Pull sono caratterizzate dalla risoluzione di una funzione di ottimizzazione. In ognuna di esse, a seconda dal tipo di politica in esame, è presente un set di vincoli sulla funzione obbiettivo.
Tutti i sistemi in partenza sono vuoti e le equazioni valgono sia nella fase di transitorio che nella fase di comportamento a regime del sistema produttivo.

2.5.1.2 Modello di simulazione per il Kanban Control System - KCS

Nei sistemi KCS una parte completata allo stage J-1 è autorizzata ad entrare allo stage successivo, solo se è disponibile una autorizzazione per lo stage J.
L'unico vincolo temporale presente sul sistema è che le parti devono giungere ad uno stage, venire processate e successivamente rilasciate e trasferite ad uno stage successivo grazie ad un'autorizzazione.
In questa sezione, viene rappresentato, a titolo di esempio, un sistema molto semplificato, caratterizzato da due fasi produttive, fig. 2.9.

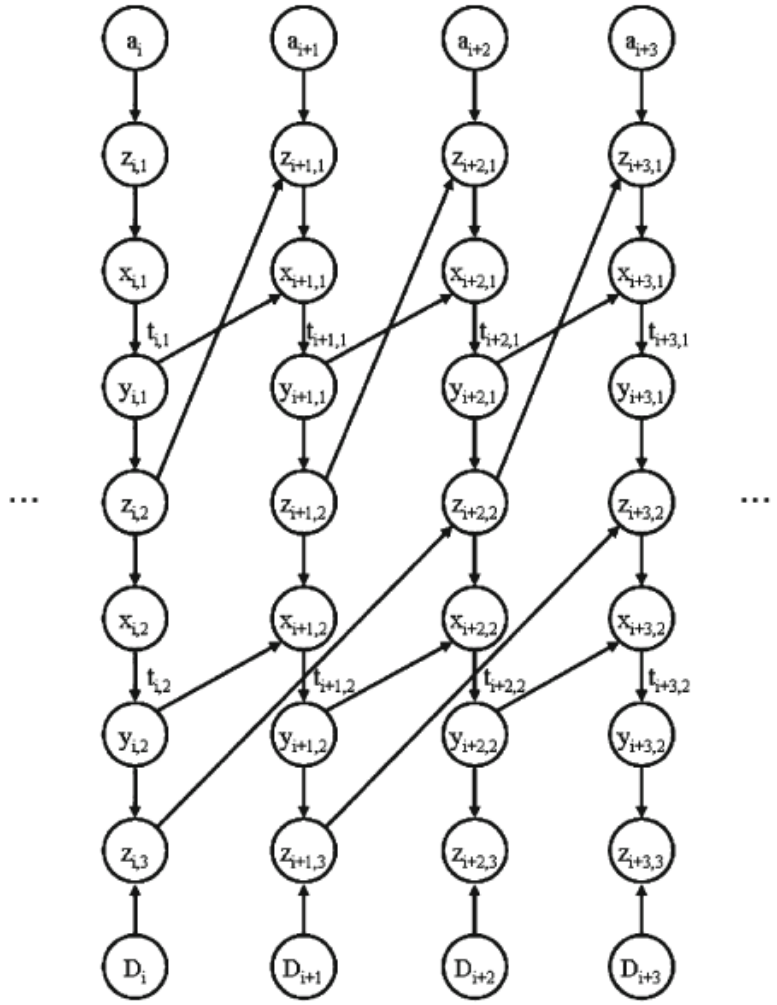

Figura 2.9. Rappresentazione grafica di un sistema KCS a due stage con K_1 =1 e K_2 = 2.

Si è considerato un sistema costituito da due stazioni, ognuna della quali è dotata di un certo numero di kanban (K_1 = 1 and K_2 = 2).
La prima stazione è caratterizzata dall'arrivo delle materie prime che vengono trasferite da un buffer di raw material alla stazione stessa.

Nella rappresentazione grafica, il trasferimento delle materie grezze da a_i alla prima stazione avviene grazie a $Z_{i,1}$.
Questo rappresenta un primo vincolo per il sistema, perché deve giungere prima la parte grezza e poi, dopo il trasferimento alla prima stazione, può essere processata.
Successivamente la parte completata sulla stazione 1 viene trasferita alla stazione 2 sempre tramite $Z_{i,2}$.
La rappresentazione grafica ci premette di notare i vincoli del sistema, fig. 2.9.
Ogni freccia rappresenta un vincolo, cioè le precedenze con le quali devono avvenire gli eventi che sono rappresentati dai cerchi.
Viene ricavata una funzione obbiettivo e i vincoli che simulano il comportamento del sistema.
I vincoli dipendono dal tipo di politica che si intende attuare: in questo caso la politica KANBAN.
Una volta osservata lo schema con il funzionamento del sistema (fig. 2.9) è possibile passare alla visualizzazione delle equazioni e disequazioni che regolano il modello P.L. di simulazione.
La prima rappresenta la funzione obbiettivo e le altre i vincoli per KCS [14].
Il set è il seguente:

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j} \right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \tag{2.1}$$

Con i vincoli :

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \tag{2.2}$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \tag{2.3}$$

$$Z_{i,j} \geq a_i \qquad \forall i \tag{2.4}$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \tag{2.5}$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{2.6}$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \tag{2.7}$$

$$Z_{i,j} - Z_{i-k_j,j+1} \geq 0 \qquad i = 1 + k_j, \dots, N;\ \forall j \tag{2.8}$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{2.9}$$

L'equazione (2.1) rappresenta la funzione obbiettivo. Come nel caso di qualsiasi simulazione di sistema si tende a minimizzare il valore del della somma dei tempi di inizio, di fine e di trasferimento di ogni singola parte che attraversa il sistema stesso, Chan e Schruben [12].
Le prime due equazioni, sono utilizzate per rappresentare i vincoli tecnologici e di produzione. In particolare, l'equazione (2.2) definisce che il tempo intercorrente tra l'inizio e la fine del processamento non può essere inferiore al tempo indicato per la lavorazione del pezzo i sulla macchina j ($t_{i,j}$). Infine, l'equazione (2.3) impone che una macchina non può processare due parti differenti contemporaneamente; questo è rappresentato attraverso le frecce delle priorità che vanno da $Y_{i,j}$ e $X_{i+1,j}$ (fig. 2.9).
I vincoli che seguono, vengono utilizzati per rappresentare le politiche di controllo. Il quarto vincolo, eq. (2.4), impone che il rilascio della i-esima parte al primo stage non può avvenire se non si ha materia prima a disposizione (frecce tra a_i e Z_i, fig. 2.9).
In un sistema KCS, una parte non può essere processata sullo stage j-esimo se non ha completato la lavorazione sullo stage j-1, eq. (2.5). Le frecce che riguardano tale vincolo, sono quelle tra $Y_{i,j}$ e Z_{i+1}, vedi fig. 2.9.
L'equazione (2.6) esprime il seguente vincolo: una parte non può iniziare una lavorazione in uno stage j-esimo se non è disponibile un kanban autorizzazione relativo allo stage stesso (fig. 2.9, frecce tra $Z_{i,j}$ ed $X_{i,j}$). L'equazione (2.8) indica la modalità di arretramento del flusso informativo della domanda nel sistema kanban control system. Dato uno stage j contenente un numero K_j di kanban, ad una parte i-esima è consentito entrare allo stage stesso, solo quando la parte i-K_j entra nello stage j+1 (frecce tra due eventi Z, fig. 2.9). Il trigger che avvia l'arretramento del flusso informativo della domanda è esplicitato nell'equazione (2.7), (frecce tra D_i e $Z_{i,3}$ in fig. 2.9)
Infine, tutte le variabili decisionali X, Y e Z non devono essere negative, eq. (2.9).
Il modello presenta un numero di variabili e di vincoli che dipendono dal numero di parti, dal numero di stadi della produzione e dal numero di kanban di ogni stazione produttiva. Il sistema viene risolto con un numero di iterazioni dell'algoritmo del simplesso, funzione anch'esso delle caratteristiche del sistema.

2.5.1.3 Modello di simulazione per il Base Stock Control System – BSCS

Così come la politica Kanban , il sistema Base Stock è un tipo di controllo Pull che è caratterizzato da un solo parametro decisionale.
Questo parametro è S_j, cioè il livello di stock per ogni stage j del sistema produttivo.

Questo sistema differisce dal KCS per :

- l'assenza di cartellini autorizzazione K_j.

- La presenza di un livello di scorte S_j definito per ogni stage j del sistema produttivo. Questo fissa il livello di scorte che deve essere sempre ricostituito per garantire al sistema la prontezza di risposta alla domanda.
- Il modo in cui si propaga ,all'interno del sistema produttivo, il flusso informativo relativo alla domanda di parti i-esime. Il flusso infatti, non viaggia da stazione a stazione come nel KCS ma viaggia, come detto nei capitoli precedenti, grazie alla sincronizzazione della domanda di ogni stage j con la domanda della clientela all'ultimo stage. In poche parole, quando giunge una domanda all'ultimo stage, automaticamente la domanda viene instradata a tutti gli stage j-1 a monte, non rendendo necessario l'utilizzo di alcuna autorizzazione.

Si può notare la rappresentazione grafica (fig. 2.10) di un sistema BSCS. Come sempre le frecce indicano i vincoli e le precedenze dettate dal sistema produttivo e dal tipo di logica di controllo pull adottata per esso.

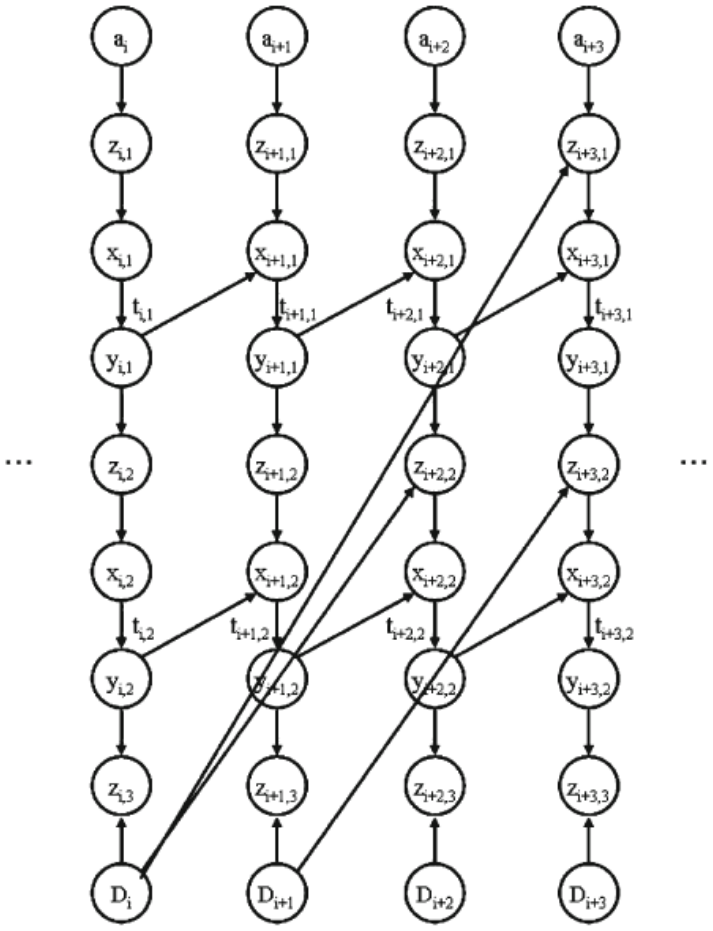

Figura 2.10. Rappresentazione grafica di un sistema BSCS a due stage con S_1=1 e S_2=2

Quindi, analizzando il set di equazioni del modello Kanban, si può notare che l'unica differenza tra Kanban e Base Stock, deriva da come si intende modellizzare il metodo di arretramento del flusso informativo della domanda.
Sostanzialmente, per simulare il comportamento del sistema BSCS, dovrà essere cambiato soltanto un vincolo tra quelli presenti nel modello per KCS (Kanban Control System) .
Prendendo in considerazione il sistema che viene riportato nell'esempio grafico (fig. 2.10), cioè un sistema che possiede un pezzo a stock nella prima stazione e due pezzi a stock nella seconda stazione, si può notare il comportamento di questa logica BSCS [14].
I vincoli di produzione, sono mantenuti, così come il vincolo di non negatività delle variabili e i vincoli generali delle politiche Pull. L'unica equazione del modello KCS che viene cambiata è la (2.8).
L'equazione (2.10) sostituisce l'equazione (2.8) del modello Kanban Control System, mentre tutte le altre sono identiche.

$$Z_{i,j} \geq D_{i-\sum_{h=j}^{M} S_h} \qquad i = 1 + \sum_{h=j}^{M} S_h \ , \dots , N; \ \ \forall j \qquad (2.10)$$

Questa equazione cambia il metodo di propagazione del flusso informativo relativo domanda all'interno del sistema. In questo sistema infatti, differentemente dal KCS, ogni stage della produzione è collegato con l'ultimo e non con il successivo [14].
Il vincolo rappresentato dall'equazione (2.10), impone che una parte i-esima non può entrare nello stage j senza che si verifiche l'arrivo di un cliente. Nella politica Base Stock, ogni stage j-esimo deve mantenere il livello di inventory fissato (S_j). Una parte processata ad un generico stage h sarà utilizzata per la soddisfazione del cliente solo dopo essere stata completata sullo stage M. Questo può succedere solo se si è verificato un consumo completo del livello di inventory degli stage a valle di essa, che ha permesso alla parte stessa di procedere in avanti [14]. Quindi l'arrivo dell'i-esimo cliente, inizializza la produzione di una parte su uno stage j-esimo; questa parte sarà utilizzata per soddisfare il cliente ($i = 1 + \sum_{h=j}^{M} S_h$)-esimo. Da questa considerazione è possibile ottenere l'indice assunto dalla domanda D all'interno dell'equazione (2.10).
Il modello con tutti i vincoli e la funzione obbiettivo, è inserito in appendice A.

3 Contributo della Tesi

Tenendo come riferimento i lavori di Matta e Alfieri in [14] per la simulazione di sistemi KCS e BSCS mediante lo sviluppo di modelli di programmazione matematica, in questa tesi si propongono due modelli approssimati per l'ottimizzazione dei sistemi KCS e BSCS.
I modelli proposti, sono approssimati dall'introduzione della variabile Time Buffer, presentati da Matta-Alfieri in [65]. La soluzione di tali modelli porta a ottenere i valori ottimi relativi ai parametri di controllo (K_j ed S_j) per ogni stadio della produzione di un sistema manifatturiero multi-stage.

All'interno della sezione 3.1 viene sviluppato nel dettaglio il concetto di Time Buffer secondo la trattazione Matta-Alfieri [65].
Questo concetto viene poi esteso ai sistemi Pull che caratterizzano questo lavoro di Tesi (KCS e BSCS). Successivamente viene presentato anche l'algoritmo necessario per l'ottimizzazione di tali sistemi.
La successiva fase di ottimizzazione di questi modelli approssimati, sezione 3.2, ha permesso la determinazione, della configurazione dei parametri di controllo K_j ed S_j e la determinazione dei tempi di lavorazione e trasferimento delle parti ottimizzati.
Dal confronto tra i risultati della soluzione approssimata (time buffer) e di quella esatta (space buffer) sono stati calcolati i Bounds relativi al livello di kanban o base stock, per ognuno degli stage caratterizzanti il sistema produttivo, vedi sezione 3.3. Nella sezione 3.4 si descrive l'algoritmo utilizzato per ottenere la soluzione in termini di configurazione del parametri kanban e base stock.

3.1 Time Buffer

In questa sezione, partendo dai modelli di simulazione MPR introdotti nel capitolo 2, che fanno riferimento a Matta-Alfieri in [14], viene presentato il concetto di Time Buffer, introdotto da Matta-Alfieri in [65].
Questo tipo di approccio, è dapprima introdotto attraverso una trattazione generalizzata contenuta in Matta-Alfieri [65] e successivamente sviluppato per i sistemi Pull, KCS e BSCS.

3.1.1 Introduzione generale del concetto di Time Buffer

Per meglio comprendere il concetto di modello esatto e di modello approssimato con i time buffer, si fa riferimento ad sistema produttivo di esempio contenuto nella trattazione Matta-Alfieri in [65].

L'esempio in [65] è costituito da una linea di code a J stazioni separate da buffer con capacità finita (c_j) che include anche i clienti che sono in lavorazione allo stage j+1. La coda al primo stage è di capacità infinita. Vengono definiti i tempi di processamento della parte i-esima sullo stage j-esimo ($t_{i,j}$). Si ipotizza che si lavori un cliente alla volta e che lo stesso non possa subire due lavorazioni contemporaneamente. Ogni cliente i deve attendere nella coda B_{j-1} se il server dello stage j è occupato nella lavorazione di un altro cliente k (con k minore di i). In questo sistema le parti non possono essere processate su uno stage se la coda immediatamente a valle è piena (si ricorda che la coda include anche le parti che sono in lavorazione allo stage di valle). Nei server non sono possibili guasti e il tempo di trasporto è considerato infinitesimo. L'indice delle prestazioni per questo sistema è caratterizzato dal throughput ossia il numero medio di parti lavorate per unità di tempo.
Introducendo il modello esatto, il sistema viene simulato mediante un modello di programmazione lineare intera. I vincoli caratterizzeranno il comportamento del sistema e la funzione obbiettivo permette di ottenere i valori ottimi dei tempi di fine lavorazione delle parti. In questo modo, la soluzione del modello di programmazione lineare (PL) corrisponde esattamente al comportamento di un sistema ad eventi discreti. Per esempio, i valori ottimi dei tempi di fine lavorazione ricavati dal modello PL, sono identici ai tempi di fine processamento di un sistema reale. Il modello infatti, lavora le stesse sequenze di parti, con gli stessi tempi di arrivo e di servizio di un sistema reale.
Da questo esempio di modello esatto è possibile comprendere il concetto di Space Buffer ossia la capacità della coda o lo spazio disponibile tra due server adiacenti, dove i clienti attendono che si liberi il server su cui devono effettuare la lavorazione.
Come riferito in precedenza, la modellizzazione esatta permette solo la simulazione del comportamento reale del sistema produttivo.
Un modo alternativo per potere modellizzare la possibilità di attesa del cliente in una coda, è quello di utilizzare il concetto di *Time Buffer*. Questo concetto, come definito da Matta-Alfieri in [65], può essere considerato come un tempo di anticipo attraverso il quale un cliente viene spinto in avanti nella produzione, su un server, prima che esso risulti effettivamente pronto per lavorarlo. Per esempio, ipotizzando le condizioni operative del sistema proposto in precedenza, si consideri una linea composta da j server in serie con due clienti a e b che devono essere serviti, uno alla volta, nella sequenza a → b, Matta-Alfieri [65].
Viene definito con s_j, il time buffer posto tra la stazione j e j+1.
Nell'eq. (3.1), si può notare come il time buffer vincoli il tempo di inizio lavorazione del cliente b allo stage j (che viene definito $x_{b,j}$) e il tempo di fine lavorazione del cliente a allo stage j+1 (definito con $y_{a,j+1}$):

$$x_{b,j} \geq y_{a,j+1} - s_j \qquad (3.1)$$

L'eq. (3.1), afferma che il cliente b allo stage j può cominciare la propria lavorazione, s_j unità di tempo prima che il cliente a termini la lavorazione sullo stage j+1. Questo concetto è ulteriormente esplicitato all'interno della figura sottostante, fig. 3.1.

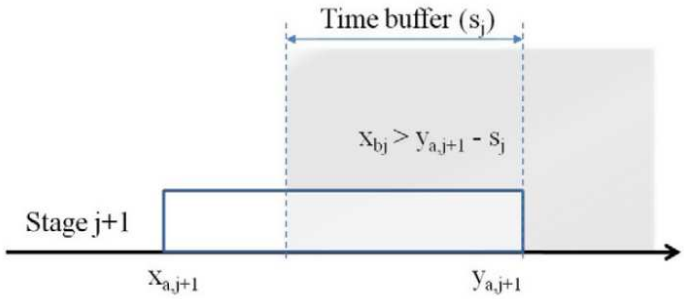

Figura 3.1. Time buffer tra gli stage j e j+1.

Dalla figura è possibile notare come vi sia, grazie al tempo di anticipo dovuto al time buffer, la possibilità di potere iniziare la lavorazione della parte b sullo stage j, anche se il server j+1 non ha ancora finito di lavorare la parte a. Si ricorda che nel modello esatto, la parte b, per potere iniziare la lavorazione sullo stage j, avrebbe dovuto attendere la fine della lavorazione della parte a sullo stage j+1.
Quando il time buffer è nullo, due stage adiacenti sono perfettamente sincronizzati. In altre parole, due stage sono completamente disaccoppiati quando il time buffer ha valore infinito [65]. La principale differenza tra time buffer e space buffer sta nel fatto che lo space buffer di capacita c è sempre in grado di ospitare un numero di parti pari a c, indipendentemente dalle condizioni del sistema. Ne consegue che il tempo speso da una parte all'interno di uno space buffer, non è limitato. Al contrario, il numero di parti che risultano inserite in un time buffer, non è definito a priori; infatti dipende dal flusso di arrivo delle parti allo stage j-1, in una certa finestra temporale (uguale al time buffer). Con il time buffer, il tempo speso dalle parti all'interno dello stesso è limitato dalla sua capacità. Da qui, si evince che, in un sistema a code, la sostituzione degli space buffer con i time buffer introduce un' approssimazione.
Il concetto di time buffer, permette di proporre una rappresentazione approssimata per sistemi multi-stage con buffer di capacità finita e tempi di produzione stocastici. L'approssimazione è dovuta al time buffer stesso, che è modellato come un vincolo tra il tempo di completamento di due parti.
Il grande vantaggio di questa formulazione è che preserva la linearità anche se viene utilizzato per l'ottimizzazione e per tale motivo può essere adottato in problemi di simulazione-ottimizzazione in modo da ridurre lo spazio delle

soluzioni. In generale, il time buffer è una variabile continua utilizzata nei modelli di ottimizzazione approssimata per sostituire una variabile intera introdotta nel modello originale di programmazione lineare.
All'interno di questo lavoro di tesi, il time buffer è stato introdotto per l'ottimizzazione dei parametri di controllo in sistemi Pull come KCS e BSCS.

3.1.2 Time buffer per sistemi Pull

I modelli presentati nel Capitolo 2, in grado di simulare la dinamica del sistema mediante programmazione lineare intera (PL intera), sono modelli esatti.
La soluzione di questi modelli, corrisponde alla reale dinamica del sistema, infatti, i tempi $X_{i,j}$,$Y_{i,j}$ e $Z_{i,j}$, (rispettivamente il tempo di inizio e fine lavorazione e il tempo di trasferimento delle parti i-esime agli stadi j-esimi) presentano gli stessi valori assunti del sistema reale, ottenendo la stessa sequenza di eventi, la stessa sequenza di domanda e gli stessi tempi di servizio di un sistema reale.
In questi modelli di simulazione esatta, il trasferimento delle parti da uno stadio al successivo, avviene quando è disponibile una risorsa/autorizzazione kanban (KCS) o quando giunge una richiesta di parti allo stadio di domanda (BSCS). Attraverso il modello di programmazione lineare esatto non è possibile ottenere il valore ottimizzato dei parametri di controllo K_j ed S_j. Di conseguenza, i modelli presenti nel capitolo 2, non sono modelli di ottimizzazione ma di simulazione.
Per ottenere i livelli ottimi di K_j e S_j, una prima possibilità potrebbe essere quella di costruire un modello matematico che incorpori i vincoli del modello di simulazione, ma avente come funzione obiettivo la minimizzazione dei costi legati al numero di kanban (caso di sistema KCS) o al livello di base stock (caso BSCS) e come vincolo addizionale il mantenimento di un livello del servizio al di sopra di un determinato valore in input. Tuttavia, le variabili decisionali di questo modello, il numero di autorizzazioni per ogni stadio e il livello del base stock per ogni stadio, sono intere. Il risultante modello di ottimizzazione sarebbe dunque intero e pertanto caratterizzato da elevata onerosità dal punto di vista del tempo e dello sforzo computazionale impiegato per la risoluzione.
Una possibilità per sviluppare il modello di ottimizzazione rimanendo nell'ambito della programmazione lineare, è di introdurre la variabile continua Time Buffer. Questo approccio è stato proposto per le linee aperte multi-stadio nel lavoro di Alfieri Matta in [65].
Il concetto di time buffer, permette di proporre una rappresentazione approssimata per sistemi multi-stage con buffer di capacità finita e tempi di produzione stocastici. L'approssimazione è dovuta al time buffer stesso, che è modellato come un vincolo tra il tempo di completamento di due parti.
Il grande vantaggio di questa formulazione è che preserva la linearità anche se viene utilizzato per l'ottimizzazione. In generale, il time buffer è una variabile continua utilizzata nei modelli di ottimizzazione approssimata per sostituire una variabile intera introdotta nel modello originale di programmazione lineare.

Questa modellizzazione approssimata, introduce delle variabili continue S_k e S_b che si riferiscono ai time buffers per il kanban e il base stock, rispettivamente.
Con queste variabili si identificano i valori di "anticipo" con cui avvengono gli eventi che simulano il trasferimento delle parti. Da qui, si può comprendere come la presenza di un "tempo di anticipo" o time buffer, su una stazione, evidenzi il bisogno del sistema di una risorsa Kanban o Base Stock. Per ogni stazione, infatti, ad ogni unità di time buffer, corrisponde una unita K_j o S_j.

3.1.2.1 Modello approssimato con Time Buffer per Kanban Control System

Di seguito viene riportato il modello approssimato con i Time Buffer per Kanban Control System. La funzione obbiettivo, per la minimizzazione del livello di kanban è la seguente :

$$Min \sum_{j=1}^{M} e_j \left[\sum_{k=1}^{N} Sk_{k,j} \right] \tag{3.2}$$

I vincoli sono :

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \tag{3.3}$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \tag{3.4}$$

$$Z_{i,j} \geq a_i \qquad \forall i \tag{3.5}$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.6}$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.7}$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \tag{3.8}$$

$$Z_{i,j} - Z_{i-k,j+1} \geq -Sk_{k,j} \qquad i = 1+k, \dots, N;\ \forall j,k \tag{3.9}$$

$$\sum_i \left(Z_{i,M+1} - D_i \right) \leq alpha \tag{3.10}$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.11}$$

La funzione obbiettivo è caratterizzata dalla minimizzazione del costo delle unità time buffer sulle stazioni j-esime (j=1,..,M).

S_{kj} rappresenta il valore del time buffer, e_j rappresenta il costo di una unità time buffer per KCS.
La matrice dei time buffer S_{kj} ha come righe il numero delle parti e come colonne il numero degli stadi della produzione. Questa matrice dipende solamente dal lag k e non dal numero delle parti i-esime, infatti, il time buffer tra la parte i e la parte i-k è lo stesso per tutte le parti i-esime.
I vincoli nelle equazioni (3.3), (3.4), (3.5), (3.6), (3.7), (3.8) e (3.11) sono comuni con i modelli di simulazione esatta presenti nel capitolo 2 e descrivono le stesse dinamiche che caratterizzano il modello esatto.
L'unico vincolo che cambia, rispetto al modello esatto, è rappresentato dall'eq. (3.9) e dall'eq. (3.10).
Il tardiness limite, eq. (3.10), viene inteso come la sommatoria dei ritardi di ogni singolo pezzo. Il ritardo di ogni parte è quantificato dalla differenza tra il tempo di arrivo del cliente e quello di scarico del pezzo al di fuori del sistema. La sommatoria del ritardo di ogni singola parte, deve essere minore o uguale ad un ritardo totale limite (total tardiness) chiamato "alpha". Dal valore del total tardiness è possibile determinare il livello di servizio fornito dal sistema produttivo.
Si può notare che la condizione (3.9) spiega la reale natura dei time buffer. Se confrontata con il vincolo (2.8) del modello di simulazione esatto, si nota subito la differenza tra lo zero (nella 2.8) e il valore del time buffer ($-S_{kj}$, nella 3.2).
Nel modello esatto (modello di simulazione, Capitolo 2) la differenza tra i due tempi di trasferimento $Z_{i,j}$ e $Z_{i\text{-}k,\, j+1}$, è maggiore o uguale a zero. Questo vincolo (fig. 3.2) deriva dal fatto che l'avanzamento delle parti avviene in presenza di autorizzazioni kanban imposte a priori. Infatti, il trasferimento della parte i allo stage j ($Z_{i,j}$) avviene o in ritardo, o in simultanea con il trasferimento della parte i-k allo stage j+1 ($Z_{i\text{-}k,\, j+1}$).

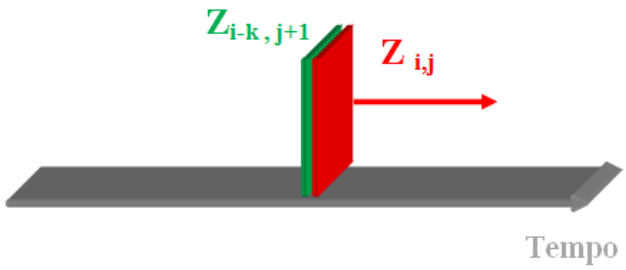

Figura 3.2. Trasferimento delle parti in un modello esatto KCS

Il ritardo o la contemporaneità del trasferimento delle parti, dipende dalla disponibilità di autorizzazioni kanban. Infatti, se sono disponibili le autorizzazioni, il trasferimento avviene in simultanea, altrimenti il sistema deve attendere la presenza di una risorsa per potere trasferire le parti da uno stadio al

successivo. Si ricorda anche che nel modello di simulazione esatto (capitolo 2), la funzione obbiettivo punta alla minimizzazione dei tempi di inizio/fine lavorazione e di trasferimento delle parti. In questi modelli di simulazione, il livello di kanban è imposto a priori e il livello di servizio viene desunto dai risultati della simulazione In altre parole, partendo da un determinato valore di K_j, il sistema si limita a simulare la dinamica del sistema, minimizzando i tempi $X_{i,j}$,$Y_{i,j}$ e $Z_{i,j}$.
Nel modello approssimato con il time buffer (modello di ottimizzazione), la funzione obbiettivo mira alla minimizzazione del costo complessivo associato alla configurazione di kanban (eq. 3.2).
Il livello delle risorse kanban viene ricavato dalla ottimizzazione stessa e non risulta imposto a priori come nel modello esatto. In aggiunta, il livello di servizio è imposto come vincolo e non è più desumibile dai risultati come nella simulazione esatta.
Il vincolo del modello approssimato che contiene il time buffer, vedi eq. (3.9), indica che l'evento $Z_{i,j}$, ossia il trasferimento di parti i allo stage j, può essere anticipato rispetto a $Z_{i-k,\ j+1}$, di un tempo pari al valore della variabile time buffer. Si ricorda che, nei sistemi che adottano una logica KCS, i trasferimenti di parti avvengono solo quando risulta disponibile una risorsa kanban. Attraverso il time buffer, il sistema ha la possibilità di trasferire le parti in anticipo, perché è in grado di ottimizzare il livello di kanban circolante, rispettando il trade-off tra la minimizzazione del costo associato a K_j o S_j e il livello di servizio imposto (fig. 3.3).
La configurazione di kanban ottima è desunta dai valori della matrice dei time buffer S_{kj}, infatti, laddove il sistema necessita di un kanban, per fare trasferire una parte in anticipo rispetto ad un'altra, il valore di tale matrice sarà diverso da zero.

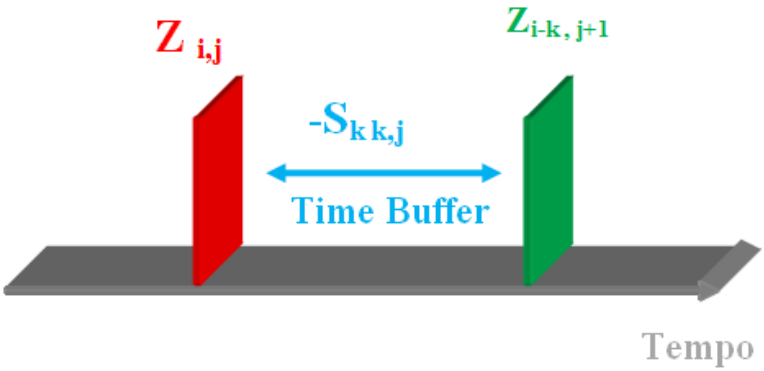

Figura 3.3. Trasferimento delle parti in un modello KCS approssimato con il time buffer

3.1.2.2 Modello approssimato con Time Buffer per Base Stock Control System

Di seguito viene riportato il modello approssimato con Time Buffer per la politica BSCS.
La funzione obbiettivo, per la minimizzazione costo associato al livello di base stock è la seguente :

$$Min \sum_{j=1}^{M} b_j \left[\sum_{k=1}^{N} Sb_{k,j} \right] \tag{3.12}$$

I vincoli sono:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \tag{3.13}$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \tag{3.14}$$

$$Z_{i,j} \geq a_i \qquad \forall i \tag{3.15}$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.16}$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.17}$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \tag{3.18}$$

$$Z_{i,j} \geq\ D_{i-g} - Sb_{gj} \qquad \forall\, i,j,\forall g < 1 \tag{3.19}$$

$$\sum_i \left(Z_{i,M+1} -\ D_i\right) \leq alpha \tag{3.20}$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.21}$$

I vincoli rappresentati nelle eq. (3.13), (3.14), (3.15), (3.16), (3.17), (3.18) e (3.21) sono comuni con il modello esatto e il modello approssimato per KCS in quanto rappresentano le dinamiche generali dei sistemi produttivi di tipo Pull.
Il vincolo con il time buffer, eq. (3.19), è proprio del modello approssimato per sistemi BSCS.
Il vincolo in eq. (3.20) si riferisce al total tardiness ed è identico a quello utilizzato nel modello approssimato per KCS, vedi eq. (3.10) .
La funzione obbiettivo mira a minimizzare il costo complessivo associato al livello di base stock del sistema produttivo. Con b_j si intende il costo di una unità time buffer.

Il vincolo sul rispetto del total tardiness limite, permette di garantire, come per il modello KCS, un determinato livello di servizio.
La matrice S_{bj} presente nel vincolo (3.19), rappresenta la matrice dei time buffer relativa al sistema BSCS.
In questo vincolo si nota come il trasferimento della parte i allo stage j ($Z_{i,j}$) può avvenire in anticipo rispetto all'arrivo della domanda $D_{i\text{-}g}$. Per meglio comprendere la natura dell'approssimazione introdotta con il time buffer, si può confrontare il comportamento del modello esatto e di quello approssimato.
Nel modello esatto (modello di simulazione) il trasferimento della parte i allo stage j ($Z_{i,j}$), avviene o in simultanea o in ritardo rispetto all'arrivo della domanda $D_{i-\sum_{h=j}^{M} S_h}$ perché il modello esatto mira alla minimizzazione dei tempi $X_{i,j}$,$Y_{i,j}$ e $Z_{i,j}$, servendosi di un livello di base stock stabilito a priori (fig. 3.3).

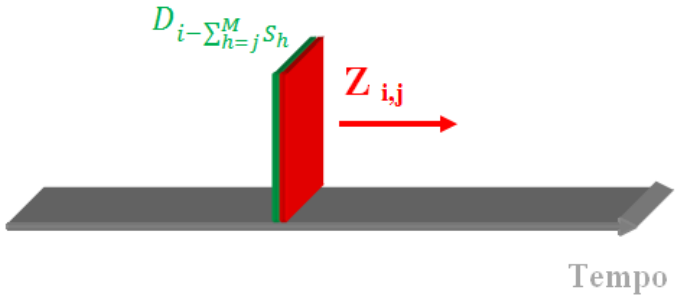

Figura 3.4. Trasferimento delle parti in un modello esatto BSCS

Si ricorda che nei sistemi Base Stock, il trasferimento verso valle delle parti avviene in presenza di una domanda finale.
Nel modello approssimato, il trasferimento della parte i sullo stage j ($Z_{i,j}$) può avvenire in anticipo rispetto all'arrivo della domanda $D_{i\text{-}g..}$ Questo anticipo (fig. 3.5) equivale al time buffer e evidenzia la presenza di una risorsa base stock. Dal time buffer, si ricava il livello di base stock necessario ad ogni stadio della produzione che viene desunto a partire dal numero di righe diverse da zero della matrice dei time buffer S_{bgj}. Come detto in precedenza, il modello approssimato ottimizza il livello di base stock gestendo il trade-off tra il costo delle unità S_j e il livello di servizio.

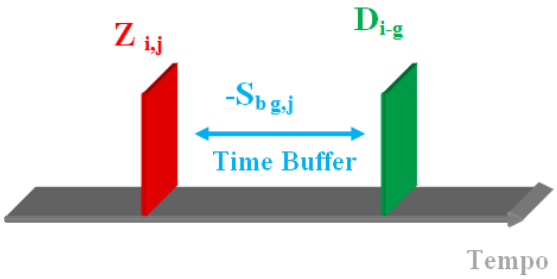

Figura 3.5. Trasferimento delle parti in un modello BSCS approssimato con il time buffer

3.2 Modelli di simulazione approssimati

Una volta ottenute la matrici dei Time Buffer S_{kj} o S_{bgj} (per KCS o BSCS), esse vengono inserite in un ulteriore modello di ottimizzazione che ha come scopo la stima delle performance del sistema caratterizzato dalla configurazione in termini di time buffer appena ricavata.
Questo modello ha come input i tempi di lavorazione, i tempi di arrivo della domanda e la matrice dei time buffer (S_{kj} o S_{bgj}, che non rappresentano più le variabili del modello). Il modello risultante perrmette di calcolare i valori ottimizzati di $X_{i,j}$, $Y_{i,j}$ e $Z_{i,j}$.

3.2.1 Modello di simulazione approssimata per il sistema KCS

Il modello seguente ha come scopo la stima delle prestazioni, in termini di tempi di inizio/fine lavorazione e di trasferimento delle parti. Una volta risolto abbiamo una soluzione in termini di time buffer ottimi.
Questa fase di ottimizzazione ha come input i tempi di lavorazione, i tempi di arrivo della domanda e la matrice dei time buffer S_{kj}. In questo caso la matrice dei time buffer non rappresenta più una delle variabili del modello. Attraverso la funzione obbiettivo si calcolano i valori ottimizzati di $X_{i,j}$, $Y_{i,j}$ e $Z_{i,j}$.
Funzione Obbiettivo (3.22):

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j} \right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \tag{3.22}$$

I vincoli sono:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \tag{3.3}$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \tag{3.4}$$

$$Z_{i,j} \geq a_i \qquad \forall i \qquad (3.5)$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.6)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.7)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (3.8)$$

$$Z_{i,j} - Z_{i-k\,,j+1} \geq -Sk_{k,j} \qquad i = 1+k, \dots, N;\ \forall j,k \qquad (3.9)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.11)$$

Come si può notare le equazioni (3.3), (3.4), (3.5), (3.6), (3.7), (3.8) e (3.11) caratterizzano il comportamento del sistema KCS e sono comuni con tutti i modelli di simulazione (capitolo 2) e ottimizzazione (capitolo 3.1) per sistemi KCS in quantodescrivono la dinamica del sistema Pull KCS.
Il vincolo con il time buffer, nell'eq. (3.9), è uguale al vincolo del modello approssimato (sezione 3.1.2.1). In questa analisi però il time buffer non risulta essere una variabile del sistema ma è dato di input, infatti, in questo modello si ottimizzano i tempi di inizio/fine lavorazione e di trasferimento delle parti che caratterizzano la configurazione di time buffer ricavata nel modello approssimato.

3.2.2 Modello di simulazione approssimata per il sistema BSCS

Il modello seguente ha come scopo la stima delle prestazioni, in termini di tempi di inizio/fine lavorazione e di trasferimento delle parti. Una volta risolto abbiamo una soluzione in termini di time buffer ottimi.
Esso ha come input i tempi di lavorazione, i tempi di arrivo della domanda e la matrice dei time buffer S_{bgj} che non rappresentano più le variabili del modello. Il modello risultante perermette di calcolare i valori ottimizzati di $X_{i,j}$, $Y_{i,j}$ e $Z_{i,j}$.
Funzione Obbiettivo (3.23):

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j} \right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \qquad (3.23)$$

I vincoli sono:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \qquad (3.13)$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \qquad (3.14)$$

$$Z_{i,j} \geq a_i \qquad \forall i \qquad (3.15)$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.16)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.17)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (3.18)$$

$$Z_{i,j} - Z_{i-k\,,j+1} \geq -Sk_{k,j} \quad i = 1+k, \dots, N;\ \forall j,k \qquad (3.19)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.21)$$

Come si può notare le equazioni (3.13), (3.14), (3.15), (3.16), (3.17), (3.18) e (3.21) caratterizzano il comportamento del sistema BSCS e sono comuni con tutti i modelli di simulazione (capitolo2) e ottimizzazione (capitolo 3) per sistemi BSCS fino a qui presentati.
Il vincolo con il time buffer, nell'eq. (3.19), è uguale al vincolo del modello approssimato (3.1.2.2). In questa analisi però il time buffer non risulta essere una variabile del sistema ma è un dato di input, infatti, in questo modello si ottimizzano i tempi di inizio/fine lavorazione e di trasferimento delle parti caratterizzati dalla configurazione di time buffer ricavata in precedenza.

3.3 Calcolo dei bounds

Una volta risolto il modello di ottimizzazione approssimato, si ottiene una soluzione in termini di time buffer ottimi.
Il problema che si affronta ora è di identificare un legame strutturale tra la soluzione ottima del problema intero originale (numero di kanban token e livello di base stock) e la soluzione in termini di time buffer.
In questa fase si fa riferimento alla relazione identificata in Alfieri e Matta in [65] dove vengono determinati i bounds, ossia, un intervallo ricavato a partire dal time buffer ottimo, all'interno del quale si ha la garanzia che giaccia la soluzione ottima del problema di ottimizzazione intero di partenza.
In questa sezione viene sviluppato il modello per il calcolo di tali limiti (bounds) nel caso di sistemi KCS e BSCS, entrambi casi che non vengono affrontati dagli autori.
Al fine di calcolare i bound è necessario conoscere i valori dei tempi di inizio attività ($X_{i,j}$) e delle matrici dei time buffer ottimi S_k ed S_b.
I tempi di inizio lavorazione utilizzati, sono quelli ricavati dai modelli di ottimizzazione approssimata (sezione 3.2) mentre le matrici dei time buffer sono quelle ricavate dai modelli approssimati (sezione 3.1).
Il risultato è la matrice dei bounds, composta da due valori per ogni stazione, che rappresentano il limite superiore e il limite inferiore del livello di K_j ed S_j.

Facendo riferimento a quanto scritto da Matta-Alfieri in [65], le due formulazioni (quella originale e quella approssimata) differiscono per come modellano il trasferimento delle parti all'interno del sistema produttivo. La formulazione approssimata ha un numero di vincoli molto più grande rispetto alla formulazione esatta perché il tempo di trasferimento ad un determinato stage è vincolato dal completamento allo stage successivo attraverso la variabile time buffer S_{kj} o S_{bj}. Le due formulazioni diventano equivalenti quando tutti i vincoli con il time buffer ad uno stage j-esimo collassano in un unico set di vincoli che coincide con i corrispettivi del modello esatto [65].
Questa condizione è ottenuta quando:

$$S_{kj} = X_{i+c_j,j} - X_{i+k,j} \;\; con \;\; i = 1, \dots, N - c_j, \;\; j = 1, \dots, J-1, \;\; k = 1, \dots, c_j, \qquad (3.24)$$

Questa equivalenza è difficile da attuare nella pratica in quanto non esiste funzione obbiettivo in grado di guidare la ricerca in questa direzione. Tuttavia, può essere usata per capire se il modello approssimato sovrastima o sottostima le performance del sistema.
Si considerino come indice per la valutazione delle prestazioni del sistema il Throughput (TH) del sistema approssimato (θ') e il TH del modello esatto (θ).
Se la formulazione esatta e la soluzione approssimata posseggono le stesse caratteristiche (tempi di arrivo e di servizio), la formulazione approssimata sovrastima le performance del sistema, (θ'>= θ), se:

$$S_{kj} > X_{i+c_j,j} - X_{i+k,j} \;\; con \;\; k = 1, \dots, c_j, i = 1, \dots, N - c_j, \;\; j = 1, \dots, J-1. \qquad (3.25)$$

e sottostima le performance (θ'<= θ), se:

$$S_{kj} < X_{i+c_j,j} - X_{i+k,j} \;\; con \;\; k = 1, \dots, c_j, i = 1, \dots, N - c_j, \;\; j = 1, \dots, J-1. \qquad (3.26)$$

Dato il modello approssimato con matrice dei time buffer S_{kj} che fornisce come soluzione i tempi $X,Y,Z_{i,j}$ e il TH θ'.
Se esiste un valore di c_j =1,…,N-1 per il quale la seguante diseguaglianza è verificata:

$$S_{kj} > X_{i+c_j,j} - X_{i+k,j} \;\; con \;\; k = 1, \dots, c_j, i = 1, \dots, N - c_j, \;\; j = 1, \dots, J-1. \qquad (3.27)$$

Per tutti i valori di k tale che $1<k<=c_j$ allora la formulazione esatta ha un TH tale che θ'>= θ.
Questo risultato permette di concludere che la formulazione approssimata con una certa configurazione di time buffer, sovrastima il TH medio di un sistema esatto con capacità c_j, avente gli stessi tempi di arrivo e di servizio.

Guardando il problema nella opposta direzione, per garantire un TH medio almeno uguale a θ', il modello esatto del sistema deve avere un valore più grande di c_j. In altre parole, c_j può rappresentare un valore limite inferiore per il livello di controllo.
Dato il modello approssimato con matrice dei time buffer S_{kj} che fornisce come soluzione i tempi X,Y,$Z_{i,j}$ e il TH θ'. Se esiste un valore di c_j =1,…,N-1 per il quale la seguante diseguaglianza è verificata:

$$S_{kj} < X_{i+c_j,j} - X_{i+k,j} \ con \ \ k = 1, \dots, c_j, i = 1, \dots, N - c_j, \ \ j = 1, \dots, J - 1. \qquad (3.28)$$

Per tutti i valori di k tale che 1<k<=c_j, allora la formulazione esatta ha un TH tale che θ>= θ'.
Analogamente a quanto scritto in precedenza, il modello approssimato con una certa configurazione di time buffer sottostima il TH medio di un modello esatto con capacità c_j avente gli stessi tempi di servizio e di interarrivo.
In altre parole, c_j, in questo caso, può rappresentare un valore limite superiore per la configurazione del livello di controllo.
Estendendo ora il concetto ai sistemi KCS e BSCS, è possibile ricavare il valore dei bounds. Come ribadito in precedenza, i tempi di inizio lavorazione (X_{ij}) sono ricavati dall'ottimizzazione del modello approssimato (sezione 3.2), mentre la matrice dei time buffer (S_{kj} per KCS o S_{bj} per BSCS) è quella ottenuta dal modello approssimato (sezione 3.1).

3.3.1 Bounds per Kanban Control System

Per quanto riguarda il Kanban, esistono i bounds se esiste un valore di $C_j^{up} = 1, \dots, N$ per il quale la seguente disequazione risulta soddisfatta:

$$S_{kj} < X_{i+c_j,j} - X_{i+k-1,j} \ con \ \ i = 1, \dots, N - C_j^{up}, j = 1, \dots, M, \ 1 < k \leq C_j^{up} \qquad (3.29)$$

e un valore di $C_j^{low} = 1, \dots, N-1$ per il quale la seguente diseguaglianza è soddisfatta:

$$S_{kj} > X_{i+c_j,j} - X_{i+k-1,j} \ con \ \ i = 1, \dots, N - C_j^{low}, j = 1, \dots, M, \ 1 < k \leq C_j^{low} \qquad (3.30)$$

Allora esiste una configurazione di space buffer che chiamiamo C_j compresa tra $C_{j\ low}$ e $C_{j\ up}$: $C_j^{low} \leq C_j \leq C_j^{up}$.
In questo caso, con C^{up} e C^{low} si intendono i limiti superiore e inferiore relativi al livello di kanban.
Questo intervallo è ricavato a partire dal time buffer ottimo, all'interno del quale si ha la garanzia che giaccia la soluzione ottima del problema di ottimizzazione intero di partenza per sistemi KSC.

3.3.2 Bounds per Base Stock Control System

Per quanto riguarda il Base Stock, esistono i bounds se esiste un valore di $C_j^{up} = 1, \dots, N-1$ per il quale la seguente disequazione risulta soddisfatta:

$$S_{bj} < X_{i+C_j,j} - X_{i+k,j} \;\; con \;\; i = 1, \dots, N - C_j^{up}, j = 1, \dots, M, \; 1 < k \leq C_j^{up} \tag{3.31}$$

e un valore di $C_j^{low} = 1, \dots, N-1$ per il quale la seguente diseguaglianza è soddisfatta:

$$S_{bj} > X_{i+C_j,j} - X_{i+k,j} \;\; con \;\; i = 1, \dots, N - C_j^{low}, j = 1, \dots, M, \; 1 < k \leq C_j^{low} \tag{3.32}$$

Allora esiste una configurazione di space buffer che chiamiamo C_j compresa tra $C_{j\,low}$ e $C_{j\,up}$: $C_j^{low} \leq C_j \leq C_j^{up}$.
In questo caso, con C^{up} e C^{low} si intendono i limiti superiore e inferiore relativi al livello di base stock.
Questo intervallo è ricavato a partire dal time buffer ottimo, all'interno del quale si ha la garanzia che giaccia la soluzione ottima del problema di ottimizzazione intero di partenza per sistemi BSCS.

3.4 Algoritmo

In questa sezione si riassume passo per passo la procedura per il calcolo del livello ottimo di Kanban e Base Stock.

3.4.1 Algoritmo per l'ottimizzazione del sistema KCS

Partendo dalle assunzioni sul sistema produttivo fatte nel capitolo 2, viene effettuata l'ottimizzazione approssimata attraverso la variabile continua time buffer. Questo modello ha come input i tempi di lavorazione delle parti su ogni stadio della produzione ($t_{i,j}$), il tempo di interarrivo della domanda (D_i) e il costo del livello di una unità K_j. In output restituisce la matrice dei time buffer (S_{kj}).
Da questa approssimazione si ricava la configurazione approssimata del livello di K_j che è desumibile dalla matrice dei time buffer, caratterizzata da un numero di righe equivalente alle parti processate dal sistema e da un numero di colonne pari agli stadi della produzione.
I valori diversi da zero della matrice dei time buffer, denotano il bisogno di una risorsa kanban da parte del sistema produttivo. Valutando il numero di righe diverse da zero per ogni stadio della produzione, è possibile stabilire il livello approssimato relativo alla configurazione di K_j (sezione 3.1).
La matrice dei time buffer relativa alla sperimentazione effettuata in questa Tesi, è presente in Appendice B.
Ottenuta la matrice dei time buffer e il livello approssimato di K_j, è possibile effettuare un' ottimizzazione in grado di donare una stima delle prestazioni del

sistema caratterizzato dalla configurazione in termini di time buffer appena ricavata.
Questo modello ha come input i tempi di lavorazione delle parti, i tempi di interarrivo della domanda e la matrice dei time buffer. Restituisce i tempi ottimi $X_{i,j}$,$Y_{i,j}$ e $Z_{i,j}$ derivanti dall'ottimizzazione di un modello approssimato con i tempi di time buffer. Risolto il modello di ottimizzazione approssimata, si è in possesso di una soluzione in termini di time buffer ottimi (sezione 3.2).
Una volta risolto il modello di ottimizzazione, vengono determinati i bound, ossia, un intervallo ricavato a partire dal time buffer ottimo, all'interno del quale si ha la garanzia che giaccia la soluzione ottima del problema di ottimizzazione intero di partenza (sezione 3.3). Calcolati i bound per il sistema KCS, si verifica che la soluzione del modello approssimato cada all'interno di questo range.

3.4.2 Algoritmo per l'ottimizzazione del sistema BSCS

Partendo dalle assunzioni sul sistema produttivo fatte nel capitolo 2, viene effettuata l'ottimizzazione approssimata attraverso la variabile continua time buffer. Questo modello ha come input i tempi di lavorazione delle parti su ogni stadio della produzione ($t_{i,j}$), il tempo di interarrivo della domanda (D_i) e il costo del livello di una unità S_j. In output restituisce la matrice dei time buffer (S_{bj}).
Da questa approssimazione si ricava la configurazione approssimata del livello di S_j che è desumibile dalla matrice dei time buffer. Anche in questo caso, come per i sistemi KCS, il valore del livello di base stock si desume dal numero di righe diverse da zero della matrice dei time buffer.
Nei sistemi echelon base stock, per passare dal valore di time buffer al valore dello space buffer occorre fare una considerazione particolare.
Si deve tenere in considerazione che i sistemi presi in esame in questa sede, sono sistemi echelon. In particolare il valore dello stock di un sistema echelon, comprende anche lo stock del livello successivo. Ecco perché il valore dei time buffer non coincide con il valore degli space buffer. L'esempio riportato di seguito (fig. 3.6) consente di chiarire la trattazione.

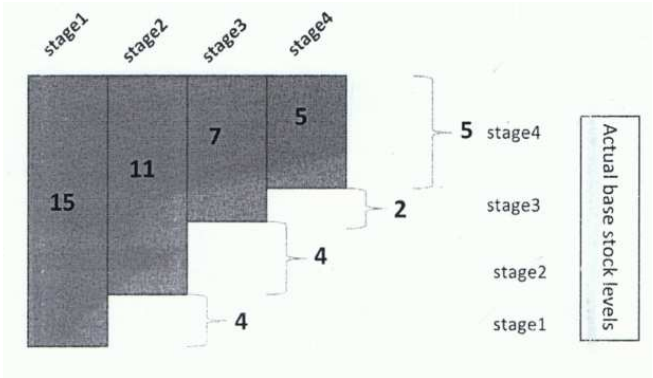

Figura 3.6. Equivalenza tra time buffers e space buffers in un sistema Base Stock.

La matrice di time buffer, relativa a questo esempio, avrà 15 numeri sulla prima colonna, 11 sulla seconda, 7 sulla terza e 5 sulla quarta.
Per delineare il livello di space buffer e quindi il livello di base stock, occorre partire dall'ultima stazione del sistema.
Partendo dallo stage 4 si desume che il livello di space buffer è 5 unità. Per lo stage 3 il livello è quello dato dalla differenza tra il livello di time buffer 7 e lo stock dello stage 4 che è 5: quindi il valore di S_3 è 2, ossia 7-5. Così si fa, arretrando, fino alla prima stazione.
In poche parole il livello di space buffer nei sistemi echelon, si calcola come differenza tra il valore del time buffer della stazione interessata e il valore del time buffer della stazione immediatamente a valle. Quindi nell'esempio 15-11 = 4 = S_1 , 11-7 = 4= S_2 , 7-5 = 2 = S_3 e 5 = S_4.
Se infatti si sommasse 5+2+4+4 si otterrebbe 15, ossia il valore del time buffer relativo allo stock echelon della prima stazione. In questi sistemi infatti ogni stazione include come livello di time buffer anche lo stock degli stage successivi.
La matrice dei time buffer relativa alla sperimentazione effettuata in questa Tesi, è presente in Appendice B.
Ottenuta la matrice dei time buffer e il livello approssimato di S_j, è possibile effettuare un' ottimizzazione in grado di donare una stima delle prestazioni del sistema caratterizzato dalla configurazione in termini di time buffer appena ricavata.
Questo modello ha come input i tempi di lavorazione delle parti, i tempi di interarrivo della domanda e la matrice dei time buffer. Restituisce i tempi ottimi $X_{i,j}$,$Y_{i,j}$ e $Z_{i,j}$ derivanti dall'ottimizzazione di un modello approssimato con i tempi di time buffer. Risolto il modello di ottimizzazione approssimata, si è in possesso di una soluzione in termini di time buffer ottimi (sezione 3.2).
Una volta risolto il modello di ottimizzazione, vengono determinati i bound, ossia, un intervallo ricavato a partire dal time buffer ottimo, all'interno del quale si ha la garanzia che giaccia la soluzione ottima del problema di ottimizzazione intero di partenza (sezione 3.3). Calcolati i bound per il sistema BSCS, si verifica che la soluzione del modello approssimato cada all'interno di questo range.

4 Risultati sperimentali

In questo capitolo vengono presentati i risultati delle sperimentazioni condotte in questo lavoro di Tesi.
In 4.1 viene presentata l'analisi delle condizioni relative alle sperimentazioni effettuate.
Nella sezione 4.2 è presente la trattazione che permette di validare i risultati attraverso un modello esatto sviluppato con il software OptQuest-Arena®.
Nella sezione 4.3 si trovano i risultati dei modelli MPR approssimati e il confronto di questi ultimi con i risultati ottenuti dall'ottimizzazione esatta effettuata con il software OptQuest-Arena®.

4.1 Analisi degli esperimenti

Per potere spiegare le caratteristiche del sistema produttivo utilizzato nelle sperimentazioni, ci si può riferire sia al comportamento del modello di programmazione lineare (Capitolo 2) che alle dinamiche dei modelli approssimati (Capitolo 3).
Il sistema produttivo che si è deciso di simulare è rappresentato da una linea manifatturiera composta da 3 stadi (fig.4.1). Su di essi viene lavorata una parte alla volta e ognuna passa da una stazione alla successiva, solo quando ha finito la propria lavorazione. A monte del sistema è presente un buffer di materie prime avente uno stock infinito in grado di fornire un'ampia disponibilità di materie prime alla linea. Tra ogni stadio della produzione è presente un buffer interoperazionale. Ogni stadio attinge dal buffer di monte le parti semilavorate e scarica al buffer di valle le parti sulle quali ha terminato la lavorazione. A valle dell'ultima stazione è presente un magazzino di parti finite. La domanda di parti finite giunge a quest'ultimo magazzino e preleva le parti destinate ai clienti.

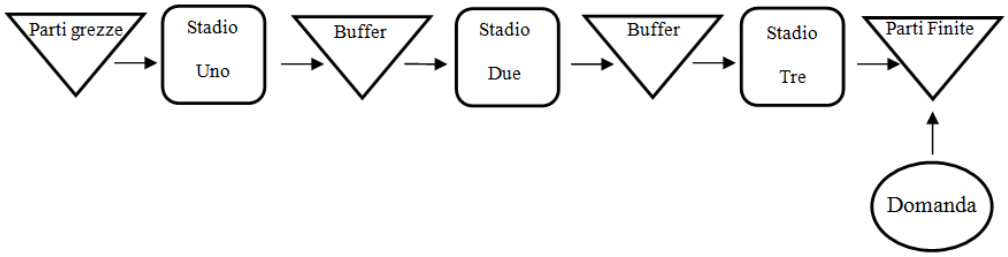

Figura 4.1. Rappresentazione del sistema produttivo considerato

Questo tipo di sistema produttivo viene analizzato, in caso di adozione di politiche Pull KCS e BSCS.
Per poter analizzare correttamente la sperimentazione, occorre focalizzare l'attenzione su alcuni elementi: i tempi di interarrivo della domanda, i tempi di lavorazione delle parti, le condizioni in cui opera il sistema (transitorio o

regime), il numero delle parti processate, i tempi di arrivo delle materie prime, i costi delle unità kanban e base stock e la definizione del livello di servizio verso la clientela
Il numero di parti è stato posto apri a 10000, in modo di consentire al sistema di potere lavorare in condizioni di regime. Le stazioni di lavoro sono J=3. I tempi di lavorazione di ogni stazione sono stati generati casualmente da una distribuzione esponenziale con λ=1. La domanda è stata generata da una Poisson di parametro λ=0,5.
Di conseguenza, i tempi di interarrivo della domanda hanno una media di 2 minuti (ossia 1/ λ). I tempi di arrivo delle parti, sono stati posti tutti uguali a 0. Questo perché si considera un sistema che ha un' ampia disponibilità di materie prime a monte della prima stazione di lavoro. Questa condizione non inficia nell'analisi dei risultati ed è spesso studiata in letteratura.
I costi delle unità kanban e base stock, sono stati presi unitari, in modo da non influenzare l'ottimizzazione dei livelli di K_j ed S_j.
La performance di riferimento per i sistemi in esame solitamente adottata è il livello di servizio (service level) espresso come (eq. (4.1)):

$$service\ level = \%\ \frac{numero\ di\ clienti\ serviti\ senza\ ritardi}{numero\ totale\ di\ clienti\ serviti} \qquad (4.1)$$

Questo concetto esprime la percentuale della domanda evasa senza ritardo. La misura di prestazione di interesse nei modelli di ottimizzazione approssimata è il total tardiness α, eq. (4.2), inteso come la somma dei ritardi nel soddisfacimento della domanda:

$$total\ tardiness = \sum_i \left(Z_{i,M+1} - D_i\right) \leq \alpha \qquad (4.2)$$

Dalla eq. (4.2) si evince che il total tardiness viene calcolato come la sommatoria del ritardo relativo a tutte le parti lavorate. Con $Z_{i,M+1}$ si intende l'arrivo delle parti al magazzino delle parti finite mentre con D_i si intende l'arrivo della domanda i-esima.
Nel caso in cui la parte giunga in anticipo questa differenza è minore di zero mentre se la parte giunge in ritardo la differenza è maggiore di zero. Dalla definizione di total tardiness è possibile comprendere il legame tra esso e il concetto di livello di servizio. I due concetti sono legati in quanto il total tardiness è la somma dei ritardi delle parti lavorate, mentre il livello di servizio è la percentuale di parti in ritardo sul totale del lavorato. Di conseguenza, fissare un livello limite per l'uno, equivale a vincolare l'altro. In queste sperimentazioni il valore del service level minimo garantito è del 98%, mentre il total tardiness limite è stato fatto variare tra 300, 200 e 100.
Il costo di una unità kanban o base stock, è stato imposto unitario. In questo modo tutti gli stadi hanno lo stesso costo di K_j ed S_j in modo da non influenzare,

in fase di ottimizzazione, la determinazione della configurazione ottima di K_j ed S_j.
In sintesi, i dati di input vengono riassunti all'interno della tabella seguente, tabella 4.1.

Tabella 4.1. Dati utilizzati per le sperimentazioni

Fattori	Valori per KCS	Valori per BSCS
Numero di parti	10000	10000
Numero di stadi	3	3
Service Level garantito	98%	98%
Total tardiness limite	300, 200, 100	300, 200, 100
Costo delle unità time buffer	Unitario per ogni stadio della produzione	Unitario per ogni stadio della produzione
Distribuzione della domanda	Esponenziale con media 2	Esponenziale con media 2
Distribuzione dei tempi di lavorazione per ogni stadio	Esponenziale con media 1	Esponenziale con media 1

In ogni esperimento vengono utilizzati gli stessi dati di input, in modo da poterne confrontare i risultati. I dati di input sono contenuti in Appendice A.
Tutti i modelli di ottimizzazione sono stati sviluppati basandosi sulle librerie IBM OPL CPLEX®. Gli esperimenti sono stati eseguiti sulle macchine presenti nei laboratori (SILAB) del Politecnico di Milano: piattaforma Windows, processore Pentium 4 a 2GHz.
I tempi di risoluzione sono riassunti nella tabella che segue:

Tabella 4.2. Tempi di risoluzione per tutte le sperimentazioni effettuate

Modello Sperimentato	Tempo di risoluzione per KCS	Tempo di risoluzione per BSCS
Modello Approssimato	3 minuti	4 minuti
Ottimizzazione Modello Approssimato	2 minuti	2 minuti
Calcolo dei Bounds	10 secondi	10 secondi
Ottimizzazione in Arena- OptQuest	36 minuti	42 minuti

4.2 Validazione dell'approccio mediante il confronto con il software OptQuest-Arena®

L'approccio matematico mediante modelli di programmazione lineare approssimati, ha permesso di ricavare il livello ottimo di kanban e base stock in sistemi KCS e BSCS (Capitolo 3). Questi livelli, essendo ricavati da un'approssimazione del comportamento del sistema, devono essere confrontati con i parametri di controllo ottenuti attraverso un modello di ottimizzazione esatta. Il confronto permette di stabilire la validità dei risultati stessi.
Un possibile approccio per ottenere valori ottimizzati di riferimento, è quello che si serve della simulazione ad eventi discreti (DES). Con essa è possibile simulare il comportamento dei sistemi che adottano le tecniche Pull e successivamente ottimizzare i livelli di controllo che caratterizzano KCS e BSCS. In questo lavoro di tesi si è scelto di eseguire la fase di simulazione e ottimizzazione attraverso l'utilizzo di un pacchetto software commerciale.
La simulazione è stata implementata attraverso Arena Rockwell®, mentre, la fase di ottimizzazione si è servita di un tool interno al software stesso (OptQuest for Arena®).
Per entrambi i sistemi sono stati realizzati i modelli di simulazione utilizzando il software Arena®. Al fine di validare i modelli di simulazione in Arena® proposti, sono state effettuate delle simulazioni di prova fornendo al modello matematico di simulazione esatta (risultato del lavoro di Matta-Alfieri in [14]) e al modello di simulazione Arena® i medesimi dati di input. Verificata la coincidenza dei valori assunti dalle variabili di interesse ($X_{i,j}$, $Y_{i,j,}$ $Z_{i,j}$) su ogni stadio e per ogni parte simulata è stato sviluppato il modello di ottimizzazione Arena® usando il software OptQuest®.
Il tool prevede la definizione di risorse, vincoli e di una funzione obbiettivo secondo la quale è possibile ottimizzare i parametri del sistema. I parametri che verranno ottimizzati sono il livello di kanban e il livello di base stock, ossia le risorse del sistema modellizzato in ambiente Arena® [66] e [67].
All'interno delle tre sezioni del tool di ottimizzazione, è possibile inserire le risorse da ottimizzare, i vincoli relativi al comportamento del sistema e la funzione obbiettivo. Il software procede, facendo variare, attraverso un range di interesse, il numero delle risorse kanban e il numero delle risorse base stock .
Sia per i sistemi KCS che per i sistemi BSCS, valgono le medesime premesse. Si impongono come risorse gli stadi della produzione e il valore delle risorse kanban e base stock j-esime. In questi modelli di ottimizzazione il vincolo è rappresentato dal rispetto di un determinato livello di servizio nei riguardi della domanda della clientela. La funzione obbiettivo, come nei modelli MPR approssimati, è rappresentata dalla minimizzazione del costo associato al livello di Kanban o Base Stock.

Per ogni configurazione provata, nel rispetto dei vincoli, l'elaboratore otterrà un risultato della funzione obbiettivo. Il sistema giudicherà come configurazione ottimale, quella che, nel rispetto del vincoli avrà ottenuto il minimo valore della funzione obbiettivo.
Operando in questo modo, è possibile che più configurazioni del livello di K_j o S_j forniscano, nel rispetto del livello di servizio imposto, lo stesso risultato in termini di funzione obbiettivo.
Ogni simulazione eseguita in Arena® è stata replicata 10 volte.
E' stato fissato lo stop automatico mentre tutte le altre opzioni sono state utilizzate come da default.
I risultati forniti dal modello OptQuest® sono stati validati attraverso il confronto con i valori disponibili in letteratura ed in particolare con quelli forniti da Di Mascolo in [37]. Il confronto, effettuato lanciando delle condizioni sperimentali pilota, ha permesso di verificare che il modello OptQuest® è in accordo con questi risultati esatti. In questo modo i modelli OptQuest® sono stati definiti come benchmark e quindi validati.
In sintesi, vista la possibilità di potere confrontare i risultati del modello approssimato con quelli di un modello validato, è stato possibile accertare l'esattezza dei modelli MPR approssimati con la variabile time buffer.

4.3 Risultati delle Sperimentazioni con i modelli MPR e calcolo dei Bounds

In questa sezione sono presentati i risultati relativi all'ottimizzazione dei modelli approssimati e al calcolo dei livelli limite (Bounds).
Per ognuna delle politiche Pull analizzate (KCS e BSCS) i risultati sono stati confrontati con il modello validato sviluppato in OptQuest®.

4.3.1 Risultati per modello KCS e calcolo dei Bounds

I risultati dei modelli approssimati con la variabile time buffer sono riassunti di seguito. Il valore è relativo ad ogni stadio della produzione al variare del total tardiness limite.
All'interno della tabella si trova il numero di cartellini kanban per ogni stadio. Si ricorda che il valore di K_j viene desunto dal numero di righe diverse da zero della matrice dei time buffer S_{kj} determinata durante il processo di ottimizzazione approssimata dei modelli MPR (sezione 3.4).
Per ogni valore di total tardiness limite è presente una configurazione del livello di kanban ottima. La matrice dei time buffer è inclusa in Appendice B.

Tabella 4.3. Risultati del modello approssimato con il time buffer - KCS

TOTAL TARDINESS LIMITE	KANBAN STADIO 1	KANBAN STADIO 2	KANBAN STADIO 3
300	2	3	7
200	3	3	7
100	2	3	8

All'aumentare di α, cioè del tardiness totale, i risultati relativi al livello di kanban, diminuiscono perché il vincolo, "rilassandosi", consente al sistema di potersi permettere ritardi via via maggiori richiedendo l'utilizzo di un numero minore di risorse kanban.
In altre parole, si può concludere che un sistema che ha un tardiness limite più elevato, garantirà un livello di servizio minore e necessiterà quindi di un valore più basso del livello di kanban.
Dalla figura 4.2 si evince che il sistema necessità di un numero maggiore di risorse kanban all'ultima stazione. Questo fatto è confermato dai riferimenti presenti in letteratura, Di Mascolo in [37]. In questo grafico è possibile notare anche l'andamento del livello di K_j in funzione di α.

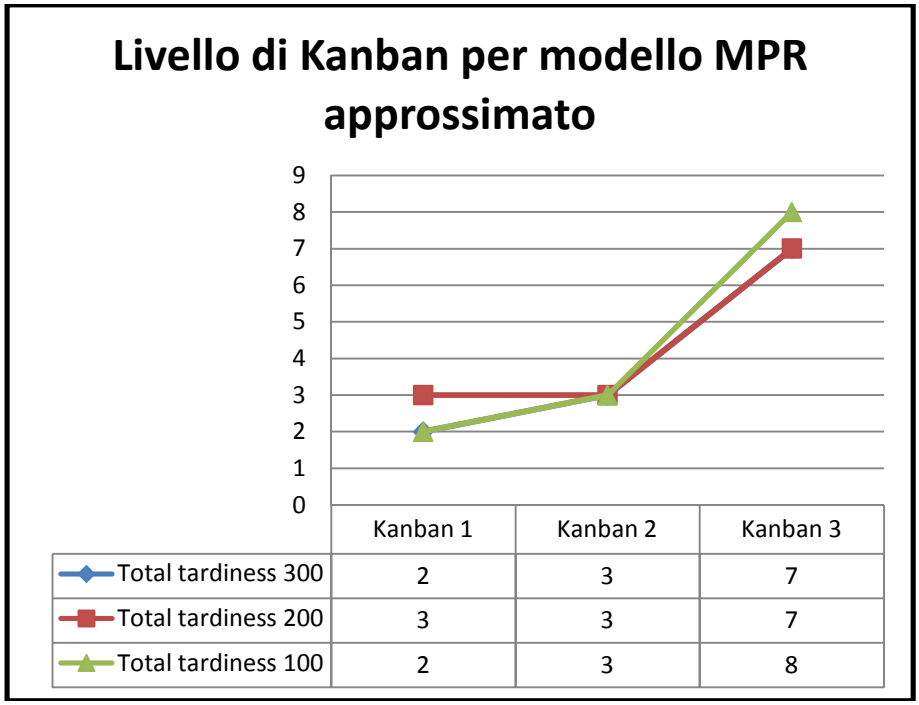

Figura 4.2. Livello di Kanban per ogni stadio, al variare di α

In figura 4.3 è possibile notare come, al diminuire del total tardiness, il livello complessivo di kanban aumenta da 300 a 200 ma rimane costante da 200 a 100, in quanto, al di sotto di un certo livello di α, il sistema non necessita più un aumento significativo della configurazione dei kanban che si attesta intorno ad un valore asintotico.

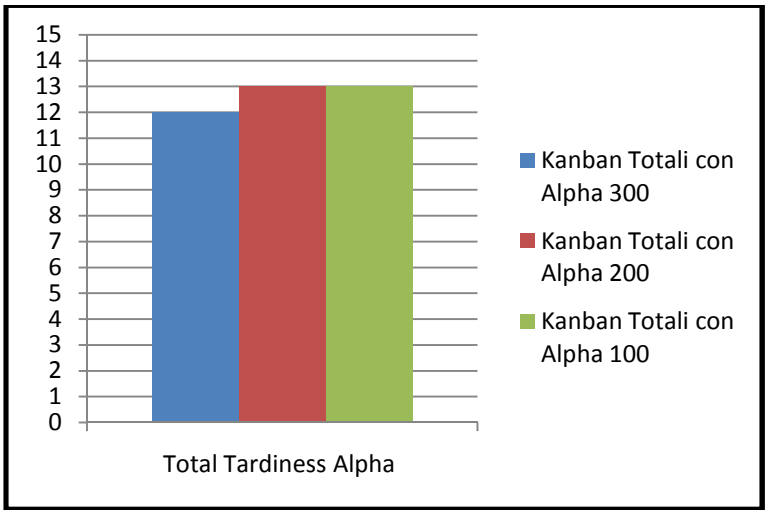

Figura 4.3. Livello complessivo di Kanban al variare del total tardiness α (Modello MPR approssimato)

La fase successiva di ottimizzazione del modello approssimato, permette di ottenere i valori di $X_{i,j}$ detti appunto "starting times". Ottenuti gli starting times e possedendo la matrice dei time buffer, è possibile calcolare i bounds agendo come mostrato nella sezione 3.3 e 3.4.
Attraverso la routine Matlab® presente nella sezione di Appendice B, è possibile i ricavare i bound per il sistema KCS.
Questi valori rappresentano il minimo e massimo numero di kanban possibili per ogni stage del sistema, ossia i limiti per la soluzione esatta (tabella 4.4). Il valore preso come soluzione del modello approssimato è relativo ad un total tardiness pari a 300 (service level 98%). Come si può notare dai risultati, del livello di Kanban si attesta vicino al limite superiore della soluzione esatta (tab. 4.4).

Tabella 4.4. Risultati del modello approssimato e bounds - KCS

	LIVELLO DI KANBAN[stage 1,2,3]
SOLUZIONE MOD. APPROSSIMATO	**[2 , 3 , 7]**
UPPER BOUND	**[3 , 4 , 8]**
LOWER BOUND	**[1 , 1 , 1]**

Per potere valutare la attendibilità dei risultati ottenuti con il modello approssimato, è necessario il confrontato con i risultati ricavati dall'ottimizzazione in ambiente OptQuest® che sono riassunti nelle seguenti tabelle. Come per il mo modello MPR approssimato, sono stati forniti i risultati per tre valori di total tardiness limite (α = 100, 200 e 300).

Risultati per KCS (total tardiness = 100) – tabella 4.5:

Tabella 4.5. Livelli ottimi di Kanban per i tre stage α=100

Livello di K_1	Livello di K_2	Livello di K_3
2	2	10
3	2	9

Risultati per KCS (total tardiness = 200) – tabella 4.6:

Tabella 4.6. Livelli ottimi di Kanban per i tre stage α=200

Livello di K_1	Livello di K_2	Livello di K_3
2	2	10
3	2	9
2	5	7

Risultati per KCS (total tardiness = 300) – tabella 4.7:

Tabella 4.7. Livelli ottimi di Kanban per i tre stage α=300

Livello di K_1	Livello di K_2	Livello di K_3
2	1	10
3	1	9
1	5	7
4	1	8
1	4	8
1	3	9

L'ottimizzatore ha restituito più di una configurazione ottima, per ogni livello di total tardiness limite. Ognuna presenta lo stesso valore di funzione obbiettivo (costo complessivo associato al livello di kanban) nel rispetto del vincolo sul total tardiness.
Se si confrontano i risultati ottenuti in OptQuest® (tabella 4.5, 4.6 e 4.7) con quelli dell'ottimizzazione approssimata (tabella 4.4), è possibile notare che il modello MPR approssimato sovrastima il livello di kanban alle prime due stazioni mentre sottostima il livello sull'ultima stazione. Questa leggera

differenza tra i due modelli, è giustificata dall'approssimazione introdotta dall'utilizzo della variabile time buffer in fase di ottimizzazione
Come si può notare dal grafico seguente, anche nell' ottimizzazione effettuata in OptQuest®, il livello di kanban cresce al diminuire del total tardiness (fig.4.4).

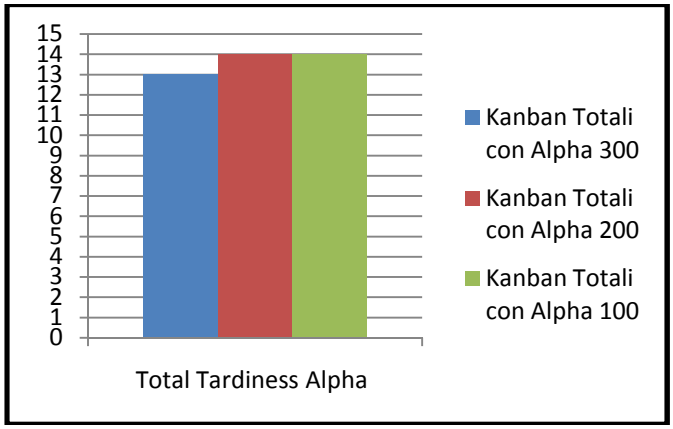

Figura 4.4. Livello complessivo di Kanban al variare del total tardiness α (Modello di ottimizzazione OptQuest-Arena®)

Come nel modello MPR approssimato, il livello di kanban cresce per α che va da 300 a 200 ma rimane costante per α che varia da 200 a 100. Alla base di questo comportamento vi è sempre la stessa motivazione fornita in precedenza. Questa analogia, rafforza in modo ulteriore i risultati forniti dal modello approssimato. Nonostante le differenze di K_j su ogni stadio, se si confronta il livello complessivo di kanban, si nota che le soluzioni offerte dai due modelli, sono molto vicine anche se i risultati del modello di programmazione lineare sottostimano di poco il livello complessivo di kanban del modello OptQuest® (fig. 4.5).

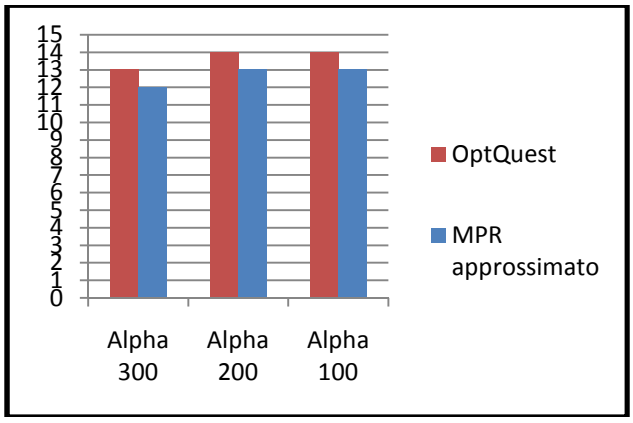

Figura 4.5. Livello di Kanban complessivo per modello MPR e modello OptQuest®, al variare del total tardiness limite α

4.3.2 Risultati per modello BSCS e calcolo dei Bounds

I dati di input e i parametri di sperimentazione sono identici a quelli utilizzati sui sistemi KCS nella sezione precedente.
I risultati dei modelli approssimati con la variabile time buffer sono relativi ad ogni stadio della produzione al variare del parametro α (total tardiness limite).
All'interno della tabella seguente si trova il numero di posizioni a stock per ogni stadio della produzione.
Si ricorda che il valore di S_j viene desunto dal numero di righe diverse da zero della matrice dei time buffer S_{bgj} e che si tratta di sistemi echelon base stock (vedi sezione 3.4).
La matrice dei time buffer è inclusa in Appendice B.
Per ogni valore di total tardiness limite è presente una configurazione del livello di base stock ottima.
I risultati per il modello approssimato Base Stock sono riassunti nella tabella 4.8.

Tabella 4.8. Risultati del modello approssimato con il time buffer - BSCS

TOTAL TARDINESS LIMITE	BASE STOCK LEVEL ALLO STADIO 1	BASE STOCK LEVEL ALLO STADIO 2	BASE STOCK LEVEL ALLO STADIO 3
300	3	3	7
200	3	3	8
100	5	3	8

I risultati (tabella 4.8), si riferiscono al livello di base stock necessario ad ogni stadio della produzione. Come già ribadito in precedenza e approfondito nella sezione 3.4, il valore presentato in tabella rappresenta il livello di stock di un sistema multi-echelon.
Come per il sistema KCS si può notare che il livello di stock diminuisce all'aumentare del total tardiness limite (α).
Anche in questo caso, il sistema produttivo, all'aumentare α necessita di un livello dei parametri di controllo inferiore consentendo al sistema di potersi permettere ritardi via via maggiori, in quanto, un sistema che ha un tardiness limite più elevato, garantirà un livello di servizio minore e necessiterà quindi di un livello più basso di base stock.
La figura 4.6 evidenzia che la maggior parte delle posizioni a stock vengono assegnate all'ultima stazione. Questo andamento è riscontrato anche il letteratura [37].

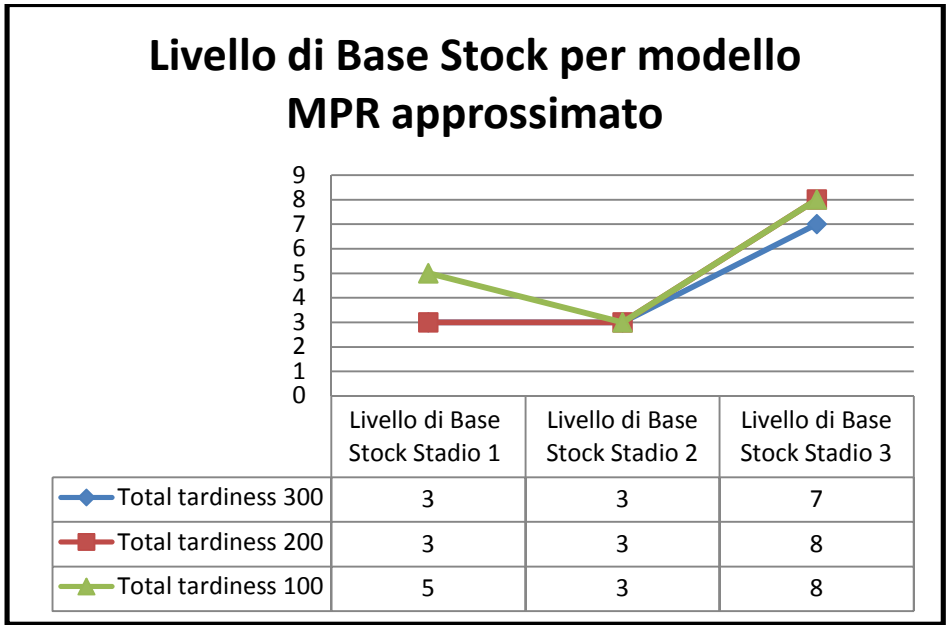

Figura 4.6. Livello di Base Stock per ogni stadio, al variare di α

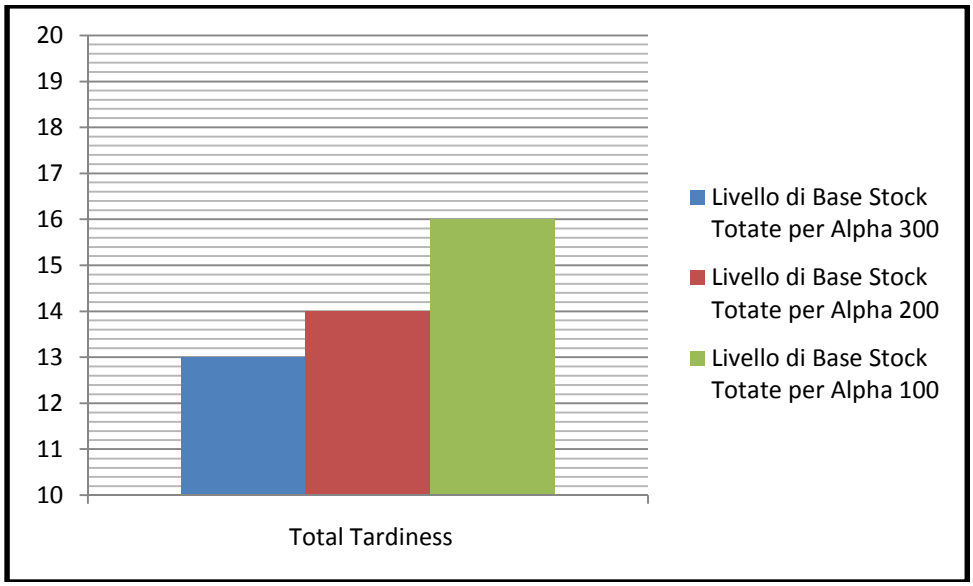

Figura 4.7. Livello complessivo di Base Stock al variare del total tardiness α

In figura 4.7 è possibile notare come, al diminuire del total tardiness, il livello di base stock complessivo (S_{tot}) aumenti. Esso aumenta molto più significativamente da 200 a 100 (S_{tot} passa da 14 a 16) che da 300 a 200 (S_{tot} passa da 13 a 14).

Dopo avere valutato i risultati del modello approssimato, si passa all'ottimizzazione dello stesso, riuscendo ad ottenere i valori di $X_{i,j}$ detti appunto "starting times", che unitamente alla matrice dei time buffer, permettono di calcolare i bounds agendo come mostrato nella sezione 3.3 e 3.4.
Attraverso la routine Matlab® presente in Appendice B, è possibile i ricavare i bound S_j. Questo valore rappresenta il minimo e massimo livello di base stock possibili per ogni stage del sistema, ossia i limiti per la soluzione esatta. Il valore preso come soluzione del modello approssimato è relativo ad un total tardiness pari a 300 (service level 98%). Come si può notare dai risultati, il livello di Base Stock si attesta vicino al limite superiore dei bounds (tab. 4.9).

Tabella 4.9. Risultati del modello approssimato e bounds – BSCS

	LIVELLO DI BASE STOCK [stage 1,2,3]
SOLUZIONE MOD. APPROSSIMATO	**[3 , 3 , 7]**
UPPER BOUND	**[3 , 3 , 8]**
LOWER BOUND	**[1 , 1 , 1]**

Per potere valutare la validità dei risultati ottenuti con il modello approssimato, è necessario il confronto con i risultati ricavati dall'ottimizzazione in ambiente OptQuest® che sono riassunti nelle seguenti tabelle. Come per il modello MPR approssimato, sono stati forniti i risultati per tre valori di total tardiness limite.
Risultati per BSCS (total tardiness = 100) – tabella 4.10:

Tabella 4.10. Livelli ottimi di Base Stock per i tre stage α=100

Livello di K_1	**Livello di K_2**	**Livello di K_3**
3	4	10

Risultati per BSCS (total tardiness = 200) – tabella 4.11:

Tabella 4.11. Livelli ottimi di Base Stock per i tre stage α=200

Livello di K_1	**Livello di K_2**	**Livello di K_3**
3	2	10
3	3	9

Risultati per BSCS (total tardiness = 300) – tabella 4.12:

Tabella 4.12. Livelli ottimi di Base Stock per i tre stage α=300

Livello di S_1	Livello di S_2	Livello di S_3
2	1	10
3	1	9
1	4	8
1	3	9

Per ottenere i risultati ottimizzati in ambiente OptQuest®, l'elaboratore ha impiegato mediamente 42 minuti.
L'ottimizzatore ha restituito più di una configurazione ottima, per ogni livello di total tardiness limite. Ognuna presenta lo stesso valore di funzione obbiettivo (costo complessivo associato al livello di base stock) garantendo il rispetto del vincolo sul total tardiness.
Se si confrontano i risultati ottenuti con OptQuest (tabella 4.10, 4.11 e 4.12) con quelli del modello approssimato (tabella 4.8), è possibile notare che, per il livello più alto di α, il modello MPR approssimato sovrastima il valore di S_j alle prime due stazioni mentre sottostima il livello sull'ultima stazione.
Per valori più bassi di α, i risultati sono ancora più vicini evidenziando comunque una sottostima del livello di S_j relativo all' ultimo stadio.
Questa lieve differenza è giustificabile dall'approssimazione introdotta nel modello ottimizzazzato con i time buffer.

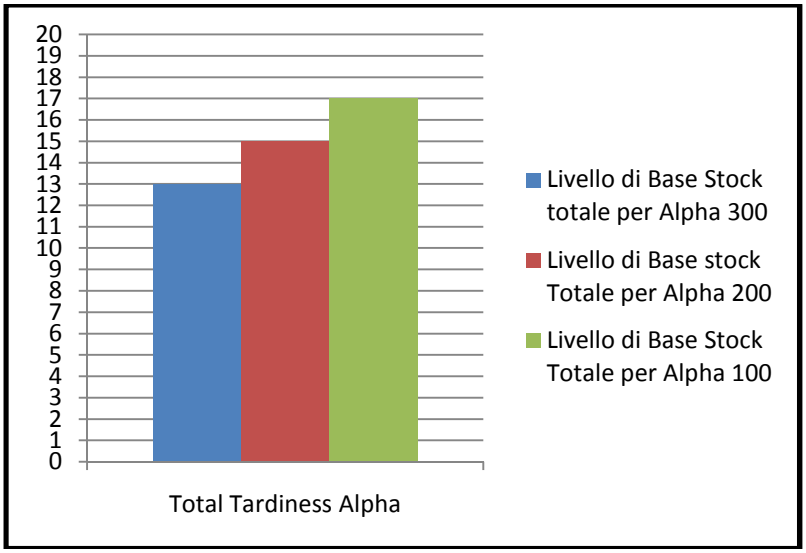

Figura 4.8. Livello complessivo di Base Stock al variare del total tardiness α (Modello di ottimizzazione OptQuest-Arena®)

Nonostante le differenze ad ogni stadio, se si confronta il livello complessivo di base stock, si nota che le soluzioni offerte dai sue modelli, sono molto vicine. I risultati del modello di programmazione lineare approssimato sottostimano il livello complessivo di base stock ottenuto con OptQuest (fig. 4.9).

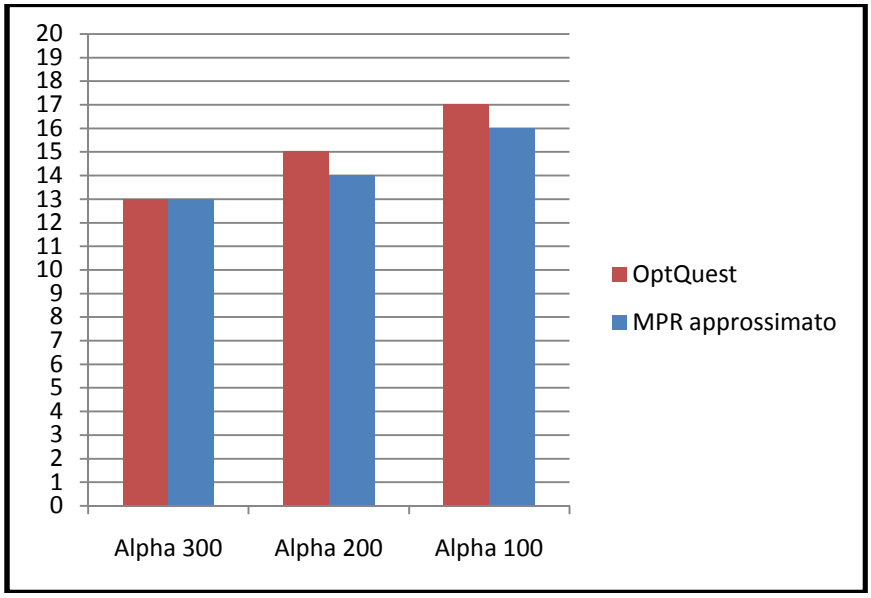

Figura 4.9. Livello di Base Stock complessivo per modello MPR e modello OptQuest®, al variare del total tardiness limite α

5 Conclusioni e Sviluppi futuri

In questo lavoro di Tesi, partendo dall'approccio di simulazione mediante Programmazione Lineare (MRP), è stato sviluppato un modello approssimato che permette l'ottimizzazione dei livelli di controllo dei sistemi Kanban Control System (KCS) e Base Stock Control System (BSCS).
Il modello presentato, attraverso la presenza della variabile continua *Time Buffer*, oltre a simulare il comportamento del sistema KCS o BSCS, permette di ottenere il livello di Kanban e Base Stock ottimizzato.
Attraverso la risoluzione del modello con i Time Buffer è possibile ricavare i risultati in termini di soluzione intera approssimata e i limiti relativi alla soluzione esatta (limite inferiore e superiore all'interno dei quali è contenuta la soluzione esatta).
I risultati ottenuti con la risoluzione del modello approssimato con la variabile time Buffer, sono stati confrontati con quelli derivanti da modelli di simulazione e ottimizzazione realizzati mediante software, precedentemente validati attraverso il confronto con le soluzioni disponibili in letteratura.
L'approccio MPR approssimato permette di ottenere una configurazione dei parametri di controllo ottima in tempi più brevi rispetto ad un software dedicato.
I risultati del modello di ottimizzazione cadono all'interno dei limiti nei quali è contenuta la soluzione esatta e ricalcano qualitativamente l'andamento dei risultati forniti dal software di ottimizzazione esatta.
Dal punto di vista quantitativo i due modelli presentano delle leggere differenze in termini di livello di K_j ed S_j derivanti dall'approssimazione introdotta con l'utilizzo della variabile time buffer. Tuttavia, esse non influenzano l'attendibilità dei risultati ottenuti col modello approssimato.
Alla luce dell'attendibilità dei risultati e della rapidità con la quale è possibile ottenerli, si può concludere che, questa tesi fornisce un nuovo approccio nel campo dell'ottimizzazione dei livelli di controllo per sistemi KCS e BSCS, utilizzando la programmazione matematica. I modelli ottenuti possono essere sfruttati come primo passo in una procedura di ottimizzazione, in quanto forniscono un algoritmo in grado di offrire una buona soluzione approssimata e robusti limiti alla soluzione esatta.

5.1 Sviluppi futuri

I risultati ottenuti in fase di ottimizzazione approssimata, hanno svelato le potenzialità della variabile time buffer applicata a logiche della produzione di tipo Pull ad un parametro di controllo: KCS e BSCS.

In futuro, per sviluppare il processo di sperimentazione, si potrebbe pensare di estendere la trattazione a sistemi Pull a due livelli di controllo come il Generalized Kanban Control System (GKCS) e l'Extended Kanban Control System (EKCS).
Questi sistemi, come citato nel Capitolo 2, controllano contemporaneamente sia il livello di Kanban che il livello di Base Stock del sistema produttivo.
Per questi due sistemi a doppio controllo, si dovrebbero costruire modelli di simulazione mediante Programmazione Lineare. Si dovrebbe introdurre la variabile Time buffer in entrambe i modelli, provvedendo successivamente, alla ottimizzazione dei modelli approssimati. Eseguita la trattazione relativa alla Programmazione Lineare, si dovrebbero implementare i modelli dei due sistemi in ambiente software. Successivamente, attraverso campagne di sperimentazione, bisognerebbe provvedere alla validazioni degli stessi sulla base delle informazioni presenti in letteratura. Una volta ottenuti i risultati dei due approcci, essi dovranno essere confrontati per valutare l'attendibilità dell'ottimizzazione dei livelli di controllo attraverso la variabile time buffer.

Bibliografia

[1] Gupta, Al Turki, An algorithm to dynamically adjust the number of Kanban in stochastic processing times and variable demand environment, Production Planning & Control, Vol.8, (1997).

[2] Sarker e Balan, Operations planning for multi stage kanban systems, European Journal of Operation Research, (1999).

[3] De Araujo, Sur l'analyse et le dimensionnement de systèmes de production gérés en Kanban, Travaux Universitaires - Thèse nouveau doctorat (1994).

[4] Clark, Scaarf, Optimal policies for a multi echelon inventory problem. Management Sci. 6, 475-490, (1960).

[5] Federgruen, Zipkin, Computational issues in an infinite-horizon, multi-echelon inventory model, Operations Research 32, 818-836 (1984).

[6] Rosling, Optimal inventory policies for assembly systems under random demands, (1989).

[7] Langenhoff, Zjim, An analytical theory of multi echelon production/distribution system, Neerlandica, (1990).

[8] Chan, Zheng, Lower bounds for multi echelon stochastic inventory systems, Management Sc. 40 1426-1443 (1994).

[9] Zjim, Van Houtum, On multi-stage production/inventory systems under stochastic demand, International Journal of Production economics, (1994).

[10] Zipkin, Gallego, Stock Positioning and performance estimation in serial production transportation systems, Manufacturing & Service Operations Management, (1999).

[11] Shang, Sheng Song, Newsvendor bounds and heuristic for optimal policies in serial supply chains, Management Science, (2003).

[12] Chan, Schruben, Optimization models of discrete event system dynamics, Operation research, 56 (5), (2008).

[13] Matta, Simulation optimization with mathematical programming representation of discrete event systems, Proceedings of the 2008 Winter Simulation Conference, pages 1393-1400 (2008).

[14] Matta, Alfieri, Mathematical programming representation of pull controlled single-product serial manufacturing systems, Journal of Intelligent Manufacturing, (2012).

[15] Nahmias, Production and Operations Analysis 6/E. McGraw-Hill, Boston, (2009).

[16] Karmarkar, Getting control of just-in-time. Harvard Business Review 67 (Sep–Oct):122–131, (1989).

[17] Zipkin PH Foundations of Inventory Management. McGraw-Hill, Boston, (2000).

[18] Liberopoulos G, Dallery Y A unified framework for pull control mechanisms in multi–stage manufacturing systems. Annals of Operations Research 93 (1–4): 325–355, (2000).

[19] Ohno K, The optimal control of just-in-time-based production and distribution systems and performance comparisons with optimized pull systems. European Journal of Operational Research 213 (1): 124–133, (2011).

[20] Karmarkar U, Getting control of just-in-time. Harvard Business Review 67 (Sep–Oct): 122–131, (1989).

[21] Buzacott JA, Shanthikumar JG Stochastic Models of Manufacturing Systems, Prentice-Hall, Englewood Cliffs, NJ, (1993).

[22] Mitra, Mitrani, Control and coordination policies for system with buffer, performance evaluation review 17(1) (1989).

[23] Gershwin, Manufacturing Systems Engineering, Prentice Hall, (1994).

[24] Bertrand JWM, Van Oijen HPG, (2002) Workload based order release and productivity: A missing link. Production Planning and Control 13 (7): 665–678.

[25] Davis, Stubitz, Configuring a kanban system using a discrete optimization of multiple stochastic responses, International Journal of Production Research, (1987).

[26] Axsäter, Rosling, Installation vs. echelon stock policies for multi-level inventory control. Management sci. 39, 1274-1280, (1989).

[27] Spearman, Woodruff, Hopp, CONWIP: a pull alternative kanban, International Journal of Production research, 28 (5), (1990).

[28] Framinan, Gonzales, The CONWIP production control system: Review and research issues, production planning and control: the management of operations, Vol. 14, issue 3, (2003).

[29] Buzacott, Queuing model of kanban and MRP controlled production systems, engineering costs and production economics, 17, 3-20, (1989).

[30] Zipkin, A kanban-like production control system: analysis of simple models, research working paper n°89-1, Graduate School of Business, Columbia University, New York, (1989).

[31] Buzacott, Shanthikumar, A general approach for coordinating production in multiple cell manufacturing systems, Production and Operation Management, 34-52, (1992).

[32] Karaesmen, Dallery, A performance comparison of pull type control mechanisms for multi stage manufacturing. International Journal of Production Economics, 59-71, (2000).

[33] Di Mascolo, Frein, Dallery, On the design of generalized kanban control systems, International Journal of Operation and Production management, 15(9), (1995).

[34] Di Mascolo, Frein, Dallery, An analytical method for performance evaluation of kanban controlled production systems, Operation Research (1996).

[35] Baynat, Di Mascolo, Frein, Dallery, A multi class approximation technique for the analysis of kanban-like control system, International Journal Of Production Research, (2001).

[36] Di Mascolo, Modeling of kanban systems using Petri nets, ORSA/TIMS Conference of flexible manufacturing systems, (1989).

[37] Di Mascolo, Duri, Frein, Comparison among three pull control policies: kanban, base stock and generalized kanban, Annals of Operation Research, (2000).

[38] Karmarkar, U.S., Kekre, S., Batching policy in kanban system. Journal of Manufacturing Systems 8 (4), 317–328, (1989).

[39] Deleersnyder J.L., Hodgson T.J.,Muller, O'Grady, P.J, Kanban Controlled Pull Systems: An Analytical Approach, Man. Sci., 35, 1079-1091, (1989).

[40] Tayur, Structural results and a heuristic for Kanban control serial lines. Management Science, 39, 1347-1368, (1993).

[41] Spearman, M. L., Customer service in pull production systems. Operations Research, 40, 948-958, (1992).

[42] Kimura O. and Terada H., Design and Analysis of Pull System: A Method of Multi-Stage Production Control, Int. J. Prod. Res., 19, 241-253, (1981).

[43] Bobbio, Roberti, Legato, Petri nets generating Markov reward models for performance analysis of degradable systems. In R. Puigjaner and D. Potier, editors, *Modeling Techniques and Tools for Computer Performance Evaluation*, pages 353-365, Plenum P.C., (1989).

[44] Philipoom P.R, Rees L.P., Taylor III B.W. and Huang P.Y., Dynamically Adjusting the Number of Kanban in a Just-in-Time Production System Using Estimated Values of Lead Time, IEE Trans., 199-207, (1987).

[45] Al Tahat, Computer supported design of flexible kanban system for multi stage multi product manufacturing, First International Industrial Engineering Conference, (2001).

[46] Al Tahat, Mukattash, design and analysis of production control scheme for kanban based JIT-environment, Journal of Franklin Institute, (2006).

[47] Wang, Sarker, optimal models for a multi stage supply chain system, controlled by kanban under just-in-time philosophy, European Journal of Operation Research, (2006).

[48] Rabbiani, Layegh, Ebrahim, Determination of number of kanban in a supply chain system via mimetic algorithm. Advances in engineering software, (2008).

[49] Chen, Guo, Lim, Multiple cross docks with inventory and time windows, (2006).

[50] Mooeni, Chang, An approximate solution in deterministic kanban system, Decision Science, (1990).

[51] Bart, Golany, Determining the number of kanban in a multi stage, multi product system, International Journal of Production research, (1991).

[52] Chang, Yih, Determining the number of kanban and lot-sizes in a generic Kanban System: a simulated annealing approach, Computer and Industrial Engineering, (1994).

[53] Shahabudden, Gopinath, Krishnaiah, Design of B-criteria kanban system using Simulated Annealing, Computer and Industrial Engineering, (2002).

[54] Kochel, P., and Nielander, U., (2000), Evolutionary Optimization of Kanban. INFORMS-KORMS-2000-Proceedings, 150-155.

[55] Widyadana, Chang, Wee, determining the optimal number of kanban in a multi-product supply chain system, Industrial Journal of System Science, (2001).

[56] Murino, Naviglio, Romano, Zoppoli, Single stage multi product kanban system, Optimization and parametric analysis, Proceedings of the 8th WSEAS International Conference on SYSTEM SCIENCE and SIMULATION in ENGINEERING, (2009).

[57] Buzacott, Price, Shanthikumar, Service level in multistage MRP and base stock controlled production systems, Conference on new directions for operation research in manufacturing, (1991).

[58] Svoronos, Zipkin, Evaluation of one –for-one replenishment policies for multi-echelon inventory systems, Management Science, (1991).

[59] Neuts, Matrix geometric solution in stochastic models, The John Hopkins University Press, (1981).

[60] Lee, Zipkin, tandem queues with planned inventories, Operation Research, (1992).

[61] Di Mascolo, Duri, Frein, On the analysis of base stock systems, in: computer integrated manufacturing and automation technology, (1996).

[62] Schruben, Mathematical programming models of discrete event system dynamics, Proceedings of the 2000 winter simulation conference, 381-385, (2000).

[63] Chan e Schruben, Properties of discrete event systems from their mathematical programming representation, Proceedings of the 2003 winter simulation conference, 496-502, (2003).

[64] Matta, Chefson, Formal properties of closed flow lines with limited buffer capacities and random processing times, Proceedings of the European simulation and modeling conference, Portugal, 190-194, (2005).

[65] Matta, Alfieri, Mathematical programming formulation for approximate simulation of multistage production systems, European journal of Operation Research, (2012).

[66] Altiok, Melamed B., Simulation Modeling and Analysis with Arena , Academic Press, (2007).

[67] Kelton W.D., Sadowski R.P., Swets N.B., Simulation with Arena , McGraw-Hill Higher Education : 1 – 668, (2004).

APPENDICE A

Nell' Appendice A vengono inclusi tutti i modelli matematici di Programmazione Lineare per KCS e BSCS.
Nella prima parte sono presenti i modelli di simulazione dei sistemi pull, inseriti nella trattazione Matta-Alfieri [14].
Nella seconda parte, sono presenti i modelli approssimati dall'introduzione della variabile Time Buffer. Da questi modelli si ricava la matrice dei time buffer e conseguentemente il valore dei livelli di controllo ottimi per Kanban Control System e Base Stock Control System.
Nella terza parte sono presenti i modelli di ottimizzazione approssimata. Da essi si ricavano gli starting times $X_{i,j}$ ottimizzati, utili al calcolo dei bounds per la soluzione esatta.

Modelli di Simulazione mediante Programmazione Lineare

Di seguito vengono sviluppati i modelli con le funzioni obbiettivo e i relativi vincoli. I Modelli di simulazione riguardano, KCS e BCS .
Si riferiscono ai modelli descritti nelle sezioni 2.5.1.2 e 2.5.1.3

Modelli di Simulazione mediante Programmazione Lineare – KANBAN CONTROL SYSTEM

Funzione Obbiettivo:

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j} \right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \tag{2.1}$$

Vincoli:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \tag{2.2}$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \tag{2.3}$$

$$Z_{i,j} \geq a_i \qquad \forall i \tag{2.4}$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (2.5)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (2.6)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (2.7)$$

$$Z_{i,j} - Z_{i-k_j,j+1} \geq 0 \qquad i = 1 + k_j, \dots, N;\ \forall j \qquad (2.8)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (2.9)$$

Modelli di Simulazione mediante Programmazione Lineare – BASE STOCK CONTROL SYSTEM

Funzione Obbiettivo:

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j} \right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \qquad (2.1)$$

Vincoli:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \qquad (2.2)$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \qquad (2.3)$$

$$Z_{i,j} \geq a_i \qquad \forall i \qquad (2.4)$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (2.5)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (2.6)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (2.7)$$

$$Z_{i,j} \geq D_{i-\sum_{h=j}^{M} S_h} \qquad i = 1 + \sum_{h=j}^{M} S_h\ , \dots, N;\ \forall j \qquad (2.10)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (2.9)$$

Modelli approssimati coi time buffer

Sono i modelli di simulazione nei quali ho introdotto la variabile continua Time Buffer. Questa variabile rende i modelli Approssimati.

Modelli Approssimati coi Time Buffer – KANBAN CONTROL SYSTEM

Funzione Obbiettivo:

$$Min \sum_{j=1}^{M} e_j \left[\sum_{k=1}^{N} Sk_{k,j} \right] \tag{3.2}$$

Vincoli:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \tag{3.3}$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \tag{3.4}$$

$$Z_{i,j} \geq a_i \qquad \forall i \tag{3.5}$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.6}$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.7}$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \tag{3.8}$$

$$Z_{i,j} - Z_{i-k,j+1} \geq -Sk_{k,j} \qquad i = 1+k, \dots, N;\ \forall j,k \tag{3.9}$$

$$\sum_i \left(Z_{i,M+1} - D_i \right) \leq alpha \tag{3.10}$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \tag{3.11}$$

Modelli Approssimati coi Time Buffer – BASE STOCK CONTROL SYSTEM

Funzione Obbiettivo:

$$Min \sum_{j=1}^{M} b_j \left[\sum_{k=1}^{N} Sb_{k,j} \right] \tag{3.12}$$

Vincoli:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \qquad (3.13)$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \qquad (3.14)$$

$$Z_{i,j} \geq a_i \qquad \forall i \qquad (3.15)$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.16)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.17)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (3.18)$$

$$Z_{i,j} \geq\ D_{i-g} - Sb_{gj} \qquad \forall\, i,j, \forall g < 1 \qquad (3.19)$$

$$\sum_i \left(Z_{i,M+1} -\ D_i\right) \leq alpha \qquad (3.20)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.21)$$

Modelli di ottimizzazione dei Modelli Approssimati

Di seguito troviamo i modelli matematici di ottimizzazione dei Modelli Approssimati coi Time Buffer.

Modello di ottimizzazione del Modello Approssimato – KANBAN CONTROL SYSTEM

Funzione Obbiettivo:

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j}\right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \qquad (3.22)$$

Vincoli:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \qquad (3.3)$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \qquad (3.4)$$

$$Z_{i,j} \geq a_i \qquad \forall i \qquad (3.5)$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.6)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.7)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (3.8)$$

$$Z_{i,j} - Z_{i-k\,,j+1} \geq -Sk_{k,j} \qquad i = 1+k, \dots, N;\ \forall j,k \qquad (3.9)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.11)$$

Modello di ottimizzazione del Modello Approssimato – BASE STOCK CONTROL SYSTEM

Funzione Obbiettivo:

$$Min \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \left(X_{i,j} + Y_{i,j} \right) + \sum_{j=1}^{M+1} Z_{i,j} \right] \qquad (3.23)$$

Vincoli:

$$Y_{i,j} - X_{i,j} \geq t_{i,j} \qquad \forall\, i,j \qquad (3.13)$$

$$X_{i+1,j} - Y_{i,j} \geq 0 \qquad i = 1, \dots, N-1;\ \forall j \qquad (3.14)$$

$$Z_{i,j} \geq a_i \qquad \forall i \qquad (3.15)$$

$$Z_{i,j+1} - Y_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.16)$$

$$X_{i,j} - Z_{i,j} \geq 0 \qquad \forall\, i,j \qquad (3.17)$$

$$Z_{i,M+1} \geq D_i \qquad \forall\, i \qquad (3.18)$$

$$Z_{i,j} \geq D_{i-g} - Sb_{gj} \qquad \forall\, i, j, \forall g < 1 \qquad (3.19)$$

$$X_{i,j}, Y_{i,j}, Z_{i,j} \geq 0 \qquad \forall\, i, j \qquad (3.21)$$

APPENDICE B

In Appendice B, sono riportati tutti i listati software dei modelli analizzati nei Capitoli 2 e 3.
In particolare sono presenti i listati per il software OPL CPLEX®, dei modelli di simulazione e di ottimizzazione approssimata inseriti nella Appendice A .
In aggiunta ci sono anche i file Matlab® per la generazione dei dati di input e per il calcolo dei bounds.
Infine ho inserito anche le matrici dei time buffer risultanti dall'ottimizzazione dei modelli Approssimati per KCS e BSCS.

Modelli di Simulazione mediante Programmazione Lineare (LISTATI OPL CPLEX®)

LISTATO OPL CPLEX® per Simulatore KCS

```
int N = ...;
int M = ...;

int k[2..M] = ...;

float D[2..N] = ...;
float a[2..N] = ...;
float t[2..N] [2..M] = ...;


//variabili

dvar float+ x[2..N] [2..M];
dvar float+ y[2..N] [2..M];
dvar float+ z[2..N] [2..M+1];

// funzione obbiettivo

minimize
      sum(i  in  2..N)sum(j  in  2..M+1)z[i][j]  +  sum(i  in
2..N)sum(j in 2..M)x[i][j]+ sum(i in 2..N)sum(j in 2..M)y[i][j];

//vincoli

subject to {
  forall ( i in 2..N , j in 2..M )
    ct2:
```

```
      y[i][j] - x[i][j] >= t[i][j];

  forall ( i in 2..N-1 , j in 2..M )
    ct3:
      x[i+1][j] - y[i][j] >= 0;

  forall ( i in 2..N )
    ct4:
        z[i][1] >= a[i];

  forall ( i in 2..N , j in 2..M )
    ct5:
      z[i][j+1] >= y[i][j];

  forall (i in 2..N , j in 2..M )
    ct6:
      x[i][j] >= z[i][j];

  forall (i in 2..N )
    ct7:
      z[i][M+1] >= D[i];

      ct8:
  forall (j in 2..M){

  forall ( i in 1+k[j]..N )

  z[i][j] - z[i-k[j]][j+1] >=0;
 }
  }
```

LISTATO OPL CPLEX® per Simulatore BSCS

```
int N = ...;
int M = ...;

int s[2..M] = ...;
int temp[2..M] = ...;

float D[2..N] = ...;
float a[2..N] = ...;
float t[2..N] [2..M] = ...;


//variabili

dvar float+ x[2..N] [2..M];
dvar float+ y[2..N] [2..M];
dvar float+ z[2..N] [2..M+1];
```

```
// funzione obbiettivo

minimize
      sum(i in 2..N)sum(j in 2..M+1)z[i][j] + sum(i in
2..N)sum(j in 2..M)x[i][j]+ sum(i in 2..N)sum(j in 2..M)y[i][j];

//vincoli

subject to {
  forall ( i in 2..N , j in 2..M )
    ct2:
      y[i][j] - x[i][j] >= t[i][j];

  forall ( i in 2..N-1 , j in 2..M )
    ct3:
      x[i+1][j] - y[i][j] >= 0;

  forall ( i in 2..N )
    ct4:
        z[i][1] >= a[i];

  forall ( i in 2..N , j in 2..M )
    ct5:
      z[i][j+1] >= y[i][j];

  forall (i in 2..N , j in 2..M )
    ct6:
      x[i][j] >= z[i][j];

  forall (i in 2..N )
    ct7:
      z[i][M+1] >= D[i];

    ct8:
   forall (j in 2..M ){
     forall (i in temp[j]..N )
       z[i][j] >= D[ i-(sum(h in j..M) s[h])] ;

   }

    }
```

Modelli Approssimati coi Time Buffer (LISTATO OPL CPLEX®)

LISTATO OPL CPLEX® per Modello approssimato per KCS

```
int N = ...;
```

```
int M = ...;

int e [2..M] = ...;
float alpha = ...;

float D[2..N] = ...;
float a[2..N] = ...;
float t[2..N] [2..M] = ...;

//variabili//

dvar float+ x[2..N] [2..M];
dvar float+ y[2..N] [2..M];
dvar float+ z[2..N] [2..M+1];

dvar float+ sk[2..N] [2..M];


dexpr float ob2 =
sum(j in 2..M)
e[j] * sum ( k in 2..N ) sk[k][j] ;

minimize

  ob2;

//sum(i in 2..N)sum(j in 2..M+1)z[i][j] + sum(i in 2..N)sum(j in
2..M)x[i][j]+
//sum(i in 2..N)sum(j in 2..M)y[i][j];

  subject to {
    forall ( i in 2..N , j in 2..M )
    ct2:
      y[i][j] - x[i][j] >= t[i][j];


  forall ( i in 2..N-1 , j in 2..M )
    ct3:
      x[i+1][j] - y[i][j] >= 0;

  forall ( i in 2..N )
    ct4:
        z[i][1] >= a[i];

  forall ( i in 2..N , j in 2..M )
    ct5:
      z[i][j+1] >= y[i][j];

  forall (i in 2..N , j in 2..M )
    ct6:
      x[i][j] >= z[i][j];
```

```
  forall (i in 2..N )
    ct7:
      z[i][M+1] >= D[i];

      ct8:
      forall (j in 2..M) {
  forall ( k in 2..50) {
    forall (i in (1+k).. N )
       z[i][j] - z[i-k][j+1] >= - sk [k][j];
        }
}
    ct9:
    sum ( i in 2..N ) ( z[i][M+1] - D[i] ) <= alpha ;

    }

   execute{
     var ofile = new IloOplOutputFile('time buffer kanban.txt');
   ofile.writeln(sk);
   }
```

LISTATO OPL CPLEX® per Modello approssimato per BSCS

```
int N = ...;
int M = ...;

int b [2..M] = ...;
int alpha= ...;


float D[2..N] = ...;
float a[2..N] = ...;
float t[2..N] [2..M] = ...;

//variabili//

dvar float+ x[2..N] [2..M];
dvar float+ y[2..N] [2..M];
dvar float+ z[2..N] [2..M+1];

dvar float+ sb[2..N] [2..M];

dexpr float ob2 =
sum(j in 2..M)
b[j] * sum (k in 2..N) sb[k][j] ;

minimize

ob2;
```

```
//sum(i in 2..N)sum(j in 2..M+1)z[i][j] + sum(i in 2..N)sum(j in
2..M)x[i][j]+
//sum(i in 2..N)sum(j in 2..M)y[i][j];

   subject to {

    forall ( i in 2..N , j in 2..M )
    ct2:
      y[i][j] - x[i][j] >= t[i][j];

  forall ( i in 2..N-1 , j in 2..M )
    ct3:
      x[i+1][j] - y[i][j] >= 0;

  forall ( i in 2..N )
    ct4:
        z[i][1] >= a[i];

  forall ( i in 2..N , j in 2..M )
    ct5:
      z[i][j+1] >= y[i][j];

  forall (i in 2..N , j in 2..M )
    ct6:
      x[i][j] >= z[i][j];

  forall (i in 2..N )
    ct7:
      z[i][M+1] - D[i] >=0;

      ct8:
      forall ( j in 2..M ){
        forall ( g in 2..50){
            forall ( i in 2..N-g )
           z[i+g][j] >= D[i] - sb [g][j];
           }
  }
       ct9:
       sum (i in 2..N) (z[i][M+1] - D[i]) <= alpha ;

}

   execute{
     var ofile = new IloOplOutputFile('time buffer base
stock.txt');
     ofile.writeln(sb);
   }
```

Modelli di Ottimizzazione del Modello approssimato (LISTATI OPL CPLEX®)

LISTATO OPL CPLEX® per Ottimizzazione del modello KCS

```
int N = ...;
 int M = ...;

 float D[2..N] = ...;
 float a[2..N] = ...;
 float t[2..N] [2..M] = ...;
 float sk[2..N] [2..M] = ...;

 dvar float+ x[2..N] [2..M];
 dvar float+ y[2..N] [2..M];
 dvar float+ z[2..N] [2..M+1];

 minimize
      sum(i in 2..N)sum(j in 2..M+1)z[i][j] + sum(i in2..N)sum(j
in 2..M)x[i][j]+ sum(i in 2..N)sum(j in 2..M)y[i][j];

//vincoli

subject to {
  forall ( i in 2..N , j in 2..M )
    ct2:
      y[i][j] - x[i][j] >= t[i][j];

  forall ( i in 2..N-1 , j in 2..M )
    ct3:
      x[i+1][j] - y[i][j] >= 0;

  forall ( i in 2..N )
    ct4:
        z[i][1] >= a[i];

  forall ( i in 2..N , j in 2..M )
    ct5:
      z[i][j+1] >= y[i][j];

  forall (i in 2..N , j in 2..M )
    ct6:
      x[i][j] >= z[i][j];

  forall (i in 2..N )
    ct7:
      z[i][M+1] >= D[i];
```

```
      ct8:
  forall (j in 2..M) {
  forall ( k in 2..50) {
    forall (i in (1+k).. N )
       z[i][j] - z[i-k][j+1] >= - sk [k][j];
        }
}
}

   execute{
     var ofile = new IloOplOutputFile('starting time
kanban.txt');
   ofile.writeln(x);
   }
```

LISTATO OPL CPLEX® per Ottimizzazione del modello BSCS

```
int N = ...;
int M = ...;

float D[2..N] = ...;
float a[2..N] = ...;
float t[2..N] [2..M] = ...;
float sb[2..N] [2..M] = ...;

 dvar float+ x[2..N] [2..M];
 dvar float+ y[2..N] [2..M];
 dvar float+ z[2..N] [2..M+1];

 minimize
      sum(i in 2..N)sum(j in 2..M+1)z[i][j] + sum(i in
2..N)sum(j in 2..M)x[i][j]+ sum(i in 2..N)sum(j in 2..M)y[i][j];

//vincoli

subject to {
  forall ( i in 2..N , j in 2..M )
    ct2:
      y[i][j] - x[i][j] >= t[i][j];

  forall ( i in 2..N-1 , j in 2..M )
    ct3:
      x[i+1][j] - y[i][j] >= 0;

  forall ( i in 2..N )
    ct4:
        z[i][1] >= a[i];

  forall ( i in 2..N , j in 2..M )
    ct5:
      z[i][j+1] >= y[i][j];
```

```
  forall (i in 2..N , j in 2..M )
    ct6:
      x[i][j] >= z[i][j];

  forall (i in 2..N )
    ct7:
      z[i][M+1] >= D[i];

      ct8:
forall ( j in 2..M ){
        forall ( g in 2..50){
            forall ( i in 2..N-g )
           z[i+g][j] >= D[i] - sb [g][j];
           }
     }
}

execute{
     var ofile = new IloOplOutputFile('starting time base
stock.txt');
     ofile.writeln(x);
   }
```

Codice per il calcolo dei bounds (LISTATO MATLAB®)

Calcolo dei Bounds per Kanban (MATLAB®)

```
clear all
clc
N=10000;
M=3;
lb=[0 0 0];
ub=[0 0 0];
x=xlsread ('foglio dati Kanban','starting times');
sk=xlsread ('foglio dati Kanban','time buffers');
count=0;
outof=0;
flag=0;
low=0;
up=0;
for j=1:M
    up=0;
    low=0;
    for c=1:N;
        up=0;
        low=1;
        for k=1:c
            up=1;
```

```
            for i=1:(N-c)

                if ( sk(k,j) > x(i+c,j)-x(i+k-1,j) )
                    up=0;
                elseif ( sk(k,j) < x(i+c,j)-x(i+k-1,j) )
                    low=0;
                end
            end
        end

                    if(low==1)
                        lb(j)=c;
                    end

                    if (up==1)
                         ub(j)=c;
                         break;
                    end
    end
end
lb
ub
```

Calcolo dei Bounds per Base Stock (MATLAB®)

```
clear all
clc
N=10000;
M=3;
lb=[0 0 0];
ub=[0 0 0];
x=xlsread ('foglio dati BaseStock','starting times');
sb=xlsread ('foglio dati BaseStock','time buffers');
count=0;
outof=0;
flag=0;
low=0;
up=0;
for j=1:M
    up=0;
    low=0;
    for c=1:(N-1);

        up=0;
        low=1;
        for k=1:c
            up=1;
```

```
            for i=1:(N-c)

                if ( sb(k,j) > x(i+c,j)-x(i+k,j) )
                    up=0;
                elseif ( sb(k,j) < x(i+c,j)-x(i+k,j) )
                    low=0;
                end
            end
        end

                    if(low==1)
                        lb(j)=c;
                    end

                    if (up==1)
                         ub(j)=c;
                         break;
                    end
    end
end
lb
ub
```

Codice per la generazione dei dati di input (LISTATO MATLAB®)

```
clc
clear
N=10000;
M=3;
a=zeros(N,1);
b=zeros(N,1);
c=zeros(N,1);
D=zeros(N,1);
start=zeros(N,1);
a = exprnd(1,N,1);
b = exprnd(1,N,1);
c = exprnd(1,N,1);
D(1,1)=15;
for i=2:N
    D(i,1)=D(i-1,1) + exprnd(2,1);
end
tempi1=(a);
tempi2=(b);
tempi3=(c);
t=[tempi1,tempi2,tempi3];
D;
xlswrite('demand.xlsx',D,'Foglio1');
xlswrite('tempo_processamento.xlsx',t,'Foglio1');
%
%
dati_domanda=D;
```

```
display(['N = ',num2str(N),';']);
display(['M = ',num2str(M),';']);
% manca la matrice con i costi dei time buffers per ogni
stage!!!
% manca il dato sul livello "alpha" di servizio!!!
display(['D = [',num2str(dati_domanda),'] ;']);
matrice_tempi=[tempi1,tempi2,tempi3];
disp('t = [');
for i=1:N
for j=1
    disp(['[',num2str(tempi1(i)),' ',num2str(tempi2(i)),'
',num2str(tempi3(i)),']']);
    j=j+1;
    end
end
disp('];');
tempi_a_inizio=start;
display(['a = [',num2str(tempi_a_inizio),'] ;']);
```

Matrici dei time Buffer (risultati di IBM OPL CPLEX®)

Nella seguente sezione , ho inserito i risultati relativi alle matrici dei time buffer al variare del parametro α (total tardiness limite). Queste matrici sono il risultato di output del Modello Approssimato coi time buffer e sono utilizzate per determinare il livello di Kj ed S_j. Esse fungono anche come input nella fase di Ottimizzazione del modello approssimato (dove si ricavano gli starting time per il calcolo dei bounds).

Matrice dei time buffer per KCS
Questa è la matrice dei time Buffer per Kanban Control System. Come detto nel Capitolo 3, il valore del livello di kanban è desunto dal numero di righe diverse da zero per ogni colonna della matrice. Ogni colonna rappresenta una stazione di lavoro. I tratteggi, indicano che la matrice continuerebbe con una serie di zeri. La matrice ha 10000 righe e 3 colonne.

Matrice dei time Buffer per alpha (total tardiness) = 300:

```
Sk =    [[2.9043   3.3782   9.311]
         [1.3465   1.8204   7.6637]
         [0        0.45633  6.2996]
         [0          0      4.7418]
         [0          0      3.184]
         [0          0      1.8584]
         [0          0      0.57806]
         [0 0 0]
         [0 0 0]
         .......
        .......
```

Matrice dei time Buffer per alpha (total tardiness) = 200:

```
Sk =    [[2.9043  3.3782   9.311]
         [1.3465  1.8204   7.6637]
         [0.5267  0.45633  6.2996]
         [0        0       4.7418]
         [0        0        3.184]
         [0        0        1.8584]
         [0        0       0.57806]
         [0 0 0]
         [0 0 0]
         .......
        .......
```

Matrice dei time Buffer per alpha (total tardiness) = 100:

```
Sk =    [[2.9043  3.3782  10.873]
         [1.3465  1.8204  9.311]
         [0       0.45633 7.6637]
         [0         0     6.2996]
         [0         0     4.7418]
         [0         0     3.184]
         [0         0     1.8584]
         [0         0     0.57806]
         [0 0 0]
         .......
        .......
```

Matrice dei time buffer per BSCS

Questa è la sezione relativa alle matrici dei time buffer per il sistema Base stock Control System per diversi valori di total tardiness limite (α).
Come affermato in precedenza, La matrice ha 10000 righe e 3 colonne che rappresentano le tre stazioni di lavoro.
I puntini di tratteggio servono ad indicare che, da quel punto in poi, la matrice è composta da zeri.
Il valore del time buffer è rappresentato dal numero di righe diverse da zero. L'unica differenza rispetto alla lettura della matrice dei time buffer per il Kanban , è che in questo caso, trattandosi di sistema echelon, dobbiamo considerare la conversione da time buffer a space buffer.
In particolare, come spiegato ampiamente nella sezione 3.4, il valore dello stock viene desunto dalla sottrazione tra il livello di time buffer della stazione di monte e quello della stazione di valle.

Matrice dei time Buffer per alpha (total tardiness) = 300:

```
Sb = [   [18.948  13.01   10.675]
         [16.551  10.96   7.7943]
         [15.146  9.1762  6.757]
         [13.215  8.3373  5.0315]
         [11.591  6.4098  3.5828]
         [10.06   5.3014  2.6047]
         [9.0188  4.0116  1.2404]
         [7.2027  2.7319    0]
         [5.611   1.6019    0]
         [3.8343  0.400     0]
         [2.9005   0        0]
         [0.90007  0        0]
         [0.13381  0        0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         .......
         .......
```

Matrice dei time Buffer per alpha (total tardiness) = 200:

```
Sb =     [20.762  15.832  12.461]
         [18.948  13.01   10.675]
         [16.551  10.96   7.7943]
         [15.146  9.1762  6.757]
         [13.215  8.3373  5.0315]
         [11.591  6.4098  3.5828]
         [10.065  5.3014  2.6047]
         [9.0188  4.0116  1.2404]
         [7.2027  2.7319       0]
         [5.611   1.6019       0]
         [3.8343  0.4000       0]
         [2.9005    0          0]
         [0.9007    0          0]
         [0.1331    0          0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         .......
         .......
```

Matrice dei time Buffer per alpha (total tardiness) = 100:

```
Sb =     [24.501  15.832  12.461]
         [22.386  13.01   10.675]
         [20.762  10.96   7.7943]
         [18.949  9.1762  6.757]
         [16.551  8.3373  5.0315]
         [15.146  6.4098  3.5828]
         [13.215  5.3014  2.6047]
         [11.591  4.0116  1.2404]
         [10.065  2.7319       0]
         [9.0188  1.6019       0]
         [7.2027  0.4000       0]
         [5.611     0          0]
         [3.8343    0          0]
         [2.9005    0          0]
         [0.9007    0          0]
         [0.133     0          0]
         [0 0 0]
         [0 0 0]
         [0 0 0]
         .......
         .......
```