
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΕΤΟΣ ΙΔΡΥΣΗΣ 1984

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Γεώργιος Λυμπερόπουλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Γεώργιος Κοζανίδης Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γεώργιος Σαχαρίδης Επίκουρος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα Καθηγητή κ. Γεώργιο Λυμπερόπουλο για την παρότρυνσή του να ασχοληθώ με το συγκεκριμένο θέμα, την άριστη συνεργασία μας και την πολύτιμη καθοδήγησή του σε κάθε στάδιο της παρούσας εργασίας. Επίσης, θα ήθελα να εκφράσω τις ευχαριστίες μου στους Καθηγητές κ. Γεώργιο Κοζανίδη και κ. Γεώργιο Σαχαρίδη που δέχτηκαν να είναι μέλη της τριμελούς επιτροπής αξιολόγησης της εργασίας. Επιπροσθέτως, είμαι ευγνώμων στη Διοίκηση της εταιρίας «Ελληνικά Γαλακτοκομεία» για την αγαστή συνεργασία και την ευγενή παραχώρηση όλων των μέσων, που χρειάστηκαν για την εκπόνηση της εργασίας. Ιδιαίτερη αναφορά αξίζει στον κ. Πίσπα Νικόλαο, διευθυντή παραγωγής της γαλακτοβιομηχανίας, η εμπειρία και η κριτική σκέψη του οποίου κατέστησαν εφικτή την εκπόνηση αυτής της εργασίας. Τέλος, θα ήθελα να ευχαριστήσω όλους τους φίλους μου που ήταν καθημερινά δίπλα μου και κατέστησαν τόσο ενδιαφέρουσα την όλη εμπειρία των σπουδών. Μα πάνω από όλα θα ήθελα να ευχαριστήσω την οικογένεια μου, τους γονείς μου Βάϊο και Μαρία και τον αδερφό μου Θωμά, για την υποστήριξη και την βοήθεια τους όλα αυτά τα χρόνια και σε αυτούς αφιερώνω την διπλωματική μου εργασία.

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΣΕ ΓΡΑΜΜΗ ΣΥΣΚΕΥΑΣΙΑΣ ΓΙΑΟΥΡΤΙΟΥ

ΠΑΠΑΠΟΣΤΟΛΟΥ ΧΡΗΣΤΟΣ

Πανεπιστήμιο Θεσσαλίας, Τμήμα Μηχανολόγων Μηχανικών 2018

Επιβλέπων Καθηγητής: Δρ. Γεώργιος Λυμπερόπουλος, Καθηγητής Διοίκησης Παραγωγής

Περίληψη

Η παρούσα εργασία, πραγματεύεται το πρόβλημα του προγραμματισμού της παραγωγής μιας βιομηχανίας τροφίμων και συγκεκριμένα το χρονοπρογραμματισμό της γραμμής παραγωγής γιαουρτιού. Στο εισαγωγικό κεφάλαιο γίνονται γνωστές βασικές έννοιες του τομέα της επιχειρησιακής έρευνας και ειδικότερα στις εφαρμογές της στη βιομηχανία. Γίνεται επίσης βιβλιογραφική ανασκόπηση όσων δημοσιεύσεων υπάρχουν πάνω στον προγραμματισμό παραγωγής της γραμμής γιαουρτιού.

Στο δεύτερο κεφάλαιο, περιγράφεται η παραγωγή γιαουρτιού με τους ειδικούς περιορισμούς της και μελετάται το σύστημα παραγωγής μίας υπάρχουσας γαλακτοβιομηχανίας στην Ελλάδα.

Στη συνέχεια της εργασίας, παρουσιάζονται δύο μοντέλα μεικτού ακέραιου γραμμικού προγραμματισμού (MILP) για το πρόβλημα σχεδιασμού και προγραμματισμού του σταδίου συσκευασίας της παραγωγής γιαουρτιού. Στο τρίτο κεφάλαιο εξετάζεται το πρόβλημα του χρονοπρογραμματισμού σε μία μονή γραμμή παραγωγής (single machine) και παρουσιάζονται τα αποτελέσματα από την επίλυση μέσω προγραμματιστικού περιβάλλοντος Cplex. Στο τέταρτο κεφάλαιο γίνεται η επέκταση σε παραγωγή με παράλληλες μηχανές (parallel machines).

Η επίλυση του προβλήματος μας προσφέρει τη βέλτιστη κατανομή και ποσότητα παραγωγής κάθε προϊόντος μέσα στο χρονικό ορίζοντα του προγραμματισμού. Οι απαντήσεις αυτές είναι οι βέλτιστες για την ελαχιστοποίηση της αντικειμενικής συνάρτησης που έχει θέσει η γαλακτοβιομηχανία ως στόχο, που είναι συγκεκριμένα η ελαχιστοποίηση του συνολικού κόστους παραγωγής. Τέλος, υπάρχουν τα συμπερασματικά σχόλια, όσον αφορά την εφικτή λύση, και το παράρτημα με τα δεδομένα και την κωδικοποίηση των μοντέλων.

Λέξεις κλειδιά: Επιχειρησιακή έρευνα, λήψη αποφάσεων, βιομηχανία γάλακτος, παραγωγή γιαουρτιού, χρονικός προγραμματισμός, βελτιστοποίηση, μαθηματική μοντελοποίηση.

Abstract

This paper addresses the problem of production scheduling of a food industry and in particular the scheduling of the yoghurt production line. The introductory chapter introduces basic concepts in the field of business research and in particular its applications in industry. There is also a bibliographic review of the publications that exist on the yoghurt production plan.

The second chapter, describes the production of yoghurt with its special constraints and the production system of an existing dairy industry in Greece is being studied.

In addition, two mixed integer-linear programming (MILP) models are presented for the design and scheduling problem of the yoghurt packaging stage. Chapter three deals with the problem of scheduling in a single line and presents the results solution values using a Cplex programming environment. In chapter four there is an extension to production with parallel machines.

Solving the problem leads to the optimal distribution and quantitiy of each product within the scheduling horizon. These answers are optimal in terms of minimizing the objective function that the dairy industry has set as a goal, namely to minimize total production costs. Finally, there are some concluding remarks on the feasible solution as well as an appendix with all the data used and the programming code.

Key words: Operational research, decision making, dairy industry, yoghurt production, scheduling, optimization, mathematical modelling.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο, παρουσιάζονται πληροφορίες εισαγωγικού χαρακτήρα, που στόχο έχουν να διαμορφώσουν στον αναγνώστη μια γενική αλλά επαρκή εικόνα του αντικειμένου, που θίγει η συγκεκριμένη εργασία, παραθέτοντας μια ανασκόπηση της σχετικής με την εργασία βιβλιογραφίας και περιγράφοντας συνοπτικά τις βασικές ενότητες της εργασίας.

1.1 Περιγραφή του γενικού προβλήματος

Με κύκλο εργασιών που διαμορφώνεται στα επίπεδα των 11,7 δισεκατομμυρίων ευρώ η βιομηχανία τροφίμων αποτελεί θεμέλιο λίθο της ελληνικής μεταποιητικής βιομηχανίας και της ελληνικής οικονομίας γενικότερα. Η παγκόσμια αναγνώριση της υψηλής διατροφικής αξίας της μεσογειακής διατροφής σε συνδυασμό με το νομοθετικό πλαίσιο για την προστασία της ονομασίας προέλευσης των προϊόντων, κατέστησαν την ελληνική βιομηχανία Τροφίμων και Ποτών ένα δυναμικό, ανταγωνιστικό και εξωστρεφή τομέα με σημαντικές επενδύσεις και επιχειρηματική δραστηριότητα, τόσο στην Ελλάδα όσο και στον υπόλοιπο κόσμο. Κυρίαρχος κλάδος της ελληνικής βιομηχανίας τροφίμων αποτελεί ο τομέας των γαλακτοκομικών προϊόντων, στον οποίο περίοπτη θέση κατέχει το γιαούρτι, η κατακόρυφη αύξηση της δημοτικότητας του οποίου έχει οδηγήσει σε εκτόξευση της ζήτησης σε παγκόσμιο επίπεδο κατά τα έτη 2010-2015 ενώ περαιτέρω ετήσια αύξηση της τάξεως του 10,54% προβλέπεται για τα έτη 2017-2020.

Οι γαλακτοβιομηχανίες προκειμένου να καλύψουν τις συνεχώς αυξανόμενες ανάγκες του καταναλωτικού κοινού, διεύρυναν το φάσμα των προϊόντων τους, προσφέροντας πλέον μια ευρεία γκάμα επιλογών, που διαφέρουν σε παραμέτρους, όπως τα διατροφικά στοιχεία, η γεύση και η συσκευασία. Η ραγδαία αύξηση, λοιπόν, του όγκου και της ποικιλίας των προϊόντων, προσέθεσε πολυπλοκότητα στην παραγωγική διαδικασία με αποτέλεσμα να κρίνεται αναγκαία η ανεύρεση τρόπων ορθολογικότερης και αποτελεσματικότερης διαχείρισης τόσο των ανθρώπινων και μηχανολογικών πόρων όσο και των πρώτων υλών. Η κυριότερη λειτουργία

αποτελεσματικής διαχείρισης των πόρων στην μεταποιητική βιομηχανία των τροφίμων είναι ο προγραμματισμός της παραγωγής, εφαρμογή που ανήκει στον ευρύτερο τομέα της επιχειρησιακής έρευνας.

Αριθμός επιχειρήσεων			
Ελλάδα		**ΕΕ-28**	
Μεταποίηση (57.971 επιχ.)	100,0%	Μεταποίηση (2.088.339 επιχ.)	100,0%
Τρόφιμα	24,9%	Μεταλλικά προϊόντα	18,3%
Μεταλλικά προϊόντα	13,0%	Τρόφιμα	12,5%
Είδη ένδυσης	12,0%	Επισκευή μηχανημάτων και εξοπλισμού	9,3%
Επισκευή μηχανημάτων και εξοπλισμού	6,6%	Προϊόντα ξύλου	8,2%
Άλλοι τομείς μεταποίησης	5,6%	Άλλοι τομείς μεταποίησης	7,3%
Κύκλος εργασιών			
Ελλάδα		**ΕΕ-28**	
Μεταποίηση (€53.735 εκατ.)	100,0%	Μεταποίηση (€7.097.395 εκατ.)	100,0%
Οπτάνθρακας και προϊόντα διύλισης	32,9%	Κατασκευή μηχανοκίνητων οχημάτων	14,0%
Τρόφιμα	21,7%	Τρόφιμα	13,3%
Βασικά μέταλλα	7,9%	Κατασκευή μηχανημάτων και ειδών εξοπλισμού	9,2%
Μεταλλικά προϊόντα	5,6%	Οπτάνθρακας και προϊόντα διύλισης	7,0%
Χημικά προϊόντα	4,1%	Μεταλλικά προϊόντα	6,8%
Αξία παραγωγής			
Ελλάδα		**ΕΕ-28**	
Μεταποίηση (€51.724 εκατ.)	100,0%	Μεταποίηση (€6.526.931 εκατ.)	100,0%
Οπτάνθρακας και προϊόντα διύλισης	33,8%	Τρόφιμα	13,2%
Τρόφιμα	19,8%	Κατασκευή μηχανοκίνητων οχημάτων	12,7%
Βασικά μέταλλα	8,1%	Κατασκευή μηχανημάτων και ειδών εξοπλισμού	9,5%
Μεταλλικά προϊόντα	5,4%	Μεταλλικά προϊόντα	7,1%
Χημικά προϊόντα	4,0%	Χημικά προϊόντα	7,0%
Ακαθάριστη προστιθέμενη αξία			
Ελλάδα		**ΕΕ-28**	
Μεταποίηση (€9.960 εκατ.)	100,0%	Μεταποίηση (€1.710.000 εκατ.)	100,0%
Τρόφιμα	26,3%	Κατασκευή μηχανημάτων και ειδών εξοπλισμού	11,7%
Μεταλλικά προϊόντα	9,2%	Κατασκευή μηχανοκίνητων οχημάτων	10,6%
Βασικά μέταλλα	7,0%	Τρόφιμα	10,6%
Χημικά προϊόντα	6,9%	Μεταλλικά προϊόντα	9,8%
Προϊόντα από μη μεταλλικά ορυκτά	6,6%	Χημικά προϊόντα	6,7%
Αριθμός εργαζομένων			
Ελλάδα		**ΕΕ-28**	
Μεταποίηση (284.307 εργαζόμενοι)	100,0%	Μεταποίηση (30.039.750 εργαζόμενοι)	100,0%
Τρόφιμα	28,2%	Τρόφιμα	13,6%
Μεταλλικά προϊόντα	9,0%	Μεταλλικά προϊόντα	12,2%
Είδη ένδυσης	6,0%	Κατασκευή μηχανημάτων και ειδών εξοπλισμού	9,8%
Βασικά μέταλλα	5,3%	Κατασκευή μηχανοκίνητων οχημάτων	8,1%
Προϊόντα από μη μεταλλικά ορυκτά	5,2%	Κατασκευή από ελαστικό και πλαστικές ύλες	5,7%

Πίνακας 1: Οικονομικά στοιχεία βιομηχανίας μεταποίησης

1.2 Επιχειρησιακή Έρευνα

1.2.1 Ορισμός

Κατά τη Βρετανική Εταιρεία Επιχειρησιακής Έρευνας:

«Η Επιχειρησιακή Έρευνα είναι η εφαρμογή των μεθόδων της επιστήμης σε πολύπλοκα προβλήματα που προκύπτουν κατά την διεύθυνση και διαχείριση μεγάλων συστημάτων που αποτελούνται από φυσικά πρόσωπα, μηχανές, υλικά και κεφάλαια στην βιομηχανία, στις επιχειρήσεις και στον στρατό. Η χαρακτηριστική προσέγγιση που πρέπει να γίνει είναι να αναπτυχθεί ένα επιστημονικό μοντέλο για το σύστημα, το οποίο να περιλαμβάνει μετρήσεις για παράγοντες όπως η τύχη και το ρίσκο, το οποίο θα προβλέπει και θα συγκρίνει τα αποτελέσματα διαφόρων αποφάσεων και στρατηγικών. Ο απώτερος στόχος είναι να βοηθά την διοίκηση να καθορίζει τις πολιτικές της και τις αποφάσεις επιστημονικώς».

1.2.2 Ιστορική αναδρομή

Επισήμως, θεωρούμε πως η Επιχειρησιακή Έρευνα ξεκίνησε λίγο πριν και κατά την διάρκεια του Β’ Παγκοσμίου Πολέμου. Τότε, δημιουργήθηκαν οι πρώτες ομάδες επιχειρησιακής έρευνας από τις βρετανικές και αμερικανικές δυνάμεις, αποτελούμενες από επιστήμονες διαφόρων τομέων, με σκοπό την αντιμετώπιση και επίλυση των τακτικών και στρατηγικών προβλημάτων που προέκυπταν κατά την διάρκεια του πολέμου. Ο όρος «Operational Research» γεννήθηκε αυτή την περίοδο, καθώς ουσιαστικά σήμαινε Research in Military Operations .

Συγκεκριμένα, σημείο αναφοράς είναι οι ομάδες που ασχολήθηκαν με την αποδοτικότερη χρήση ενός νέου ραντάρ (τότε «σύστημα έγκαιρης επισήμανσης αεροσκαφών»), όπου έγινε μελέτη, όχι μόνο του συστήματος αυτού καθαυτού, των λειτουργιών του και της ορθότερης χρήσης του εξοπλισμού, αλλά επίσης μελετήθηκε και η συμπεριφορά του ανθρώπινου δυναμικού που το χειρίζεται. Θεωρείται ότι οι

συγκεκριμένες μελέτες, διαδραμάτισαν καθοριστικό ρόλο στην έκβαση της Μάχης της Αγγλίας (1940).

Η επιτυχημένη αυτή είσοδος της επιστημονικής προσέγγισης σε στρατιωτικά θέματα επέτρεψε το άνοιγμά της και σε άλλους τομείς. Η γενικότερη βιομηχανική (και όχι μόνο) ανάπτυξη που ακολούθησε τον Β΄ Παγκόσμιο Πόλεμο δημιούργησε νέες ανάγκες, καθώς η πολυπλοκότητα των προβλημάτων που προέκυπταν σε διάφορους οργανισμούς, επιχειρήσεις αλλά ακόμη και στις κυβερνήσεις, κατέστησαν αναγκαία την εξεύρεση νέων τρόπων αντιμετώπισής τους. Δεν άργησαν να συνειδητοποιήσουν, ότι τα προβλήματα που, επιτυχώς, αντιμετωπίστηκαν κατά την διάρκεια του πολέμου είναι παρόμοια, απλά σε άλλο πλαίσιο. Αυτό είχε ως αποτέλεσμα να δημιουργηθούν επιστημονικές κοινότητες και κέντρα για την Επιχειρησιακή Έρευνα, ακαδημαϊκά προγράμματα κ.λπ. Μάλιστα, πολλά εργαλεία της Επιχειρησιακής Έρευνας που χρησιμοποιούνται ευρέως και σήμερα, αναπτύχθηκαν επαρκώς εκείνη την εποχή.

Η Επιχειρησιακή Έρευνα συνέχισε να αναπτύσσεται τις επόμενες δεκαετίες και με την δημιουργία των ηλεκτρονικών υπολογιστών αυτή η εξέλιξή της ήταν ραγδαία. Το γιατί, είναι προφανές! Η Επιχειρησιακή Έρευνα προϋποθέτει μεγάλη ποσότητα υπολογισμών, κάτι που οι υπολογιστές μπορούν να κάνουν τάχιστα, αντίθετα με τον άνθρωπο. Από το 1980 και μετά, δημιουργήθηκε και αντίστοιχο software, γεγονός, που σε συνδυασμό με την εξάπλωση των οικιακών ηλεκτρονικών υπολογιστών, επέτρεψε σε κάθε άνθρωπο να έχει πρόσβαση στην επιστήμη αυτή. Σήμερα, προγράμματα (software) όπου βοηθούν στην επίλυση προβλημάτων επιχειρησιακής έρευνας υπάρχουν σε κάθε υπολογιστή, ακόμη και αν δεν το γνωρίζουμε ή δεν τα χρησιμοποιούμε.

1.2.3 Μέθοδοι της επιχειρησιακής έρευνας

Οι μαθηματικές μέθοδοι αποτελούν ένα βασικό κομμάτι για την επίλυση των προβλημάτων της επιχειρησιακής έρευνας. Όμως, βρίσκονται στο τέλος της μελέτης του προβλήματος. Συχνά, προαπαιτείται μία ολόκληρη διαδικασία - αλληλουχία φάσεων για κάθε πρόβλημα επιχειρησιακής έρευνας για να φτάσουμε στον

κατάλληλο μαθηματικό τύπο και να προχωρήσουμε στην επίλυση του. Συνοπτικά, οι συνήθεις φάσεις μίας τέτοιας μελέτης είναι:

1. Καθορισμός του προβλήματος και συλλογή των σχετικών δεδομένων.
2. Διατύπωση ενός μαθηματικού μοντέλου.
3. Ανάπτυξη υπολογιστικών αλγορίθμων (όπου αυτό είναι απαραίτητο).
4. Έλεγχος του μοντέλου και τροποποίηση του (εάν αυτό απαιτείται).
5. Εφαρμογή του μοντέλου.

Βασικό στοιχείο της επιχειρησιακής έρευνας, λοιπόν, είναι η διαμόρφωση ενός μαθηματικού μοντέλου του προβλήματος, του οποίου η λύση επιδιώκεται. Ανάλογα με τον τύπο των προβλημάτων που αντιμετωπίζονται στην πράξη, έχουν μελετηθεί διάφορα μοντέλα και έχουν αναπτυχθεί αντίστοιχες τεχνικές. Συνοπτικά, το μεγαλύτερο ποσοστό των προβλημάτων που πραγματεύεται η επιχειρησιακή έρευνα, μπορούν να επιλυθούν με τη βοήθεια των εξής κατηγοριοποιημένων μεθόδων:

- Μέθοδοι προσομοίωσης (simulation methods) - ο στόχος είναι η δημιουργία προσομοιωτών (simulators) όπου θα επιτρέπουν στον υπεύθυνο των αποφάσεων να μελετήσει την δυνατότητα βελτιώσεων και κατόπιν να ελεγχθεί η απόδοση των βελτιώσεων αυτών.
- Μέθοδοι βελτιστοποίησης (optimization methods) - ο στόχος είναι η παρουσίαση πολλών λύσεων στον υπεύθυνο αποφάσεων με έναν τρόπο αποδοτικό και αποτελεσματικό, σε ένα περιβάλλον όπου στην πραγματικότητα οι επιλογές θα είναι άπειρες. Ο τελικός στόχος είναι η εύρεση της καλύτερης δυνατής λύσης, δεδομένων κάποιων συνθηκών.
- Μέθοδοι ανάλυσης δεδομένων (data analysis methods) - όπου ο στόχος είναι η παροχή βοήθειας στον φορέα αποφάσεων στην αναγνώριση μοτίβων και συνδέσεων στα δεδομένα που παρέχονται. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη σε διάφορες εφαρμογές, όπως στον τομέα των προβλέψεων (forecasting) αλλά και σε αυτόν της εξαγωγής (προηγουμένως άγνωστης) γνώσης από τα δεδομένα που έχουμε συλλέξει.

1.2.4 Εφαρμογές επιχειρησιακής έρευνας στη Βιομηχανία-Παραγωγή

Όπως αναλύσαμε η επιχειρησιακή έρευνα ασχολείται με τη λήψη αποφάσεων σε προσδιοριστικά και πιθανολογικά συστήματα που προκύπτουν μέσα από πραγματικά προβλήματα. Έτσι, πληθώρα ζητημάτων της βιομηχανίας μπορεί να αναλυθούν και να βελτιστοποιηθούν μέσω της επιχειρησιακής έρευνας. Συνοπτικά, η επιχειρησιακή έρευνα βρίσκει εφαρμογή σε ζητήματα όπως:

- Προγραμματισμός παραγωγής
- Προγραμματισμός προμηθειών
- Έλεγχος αποθεμάτων πρώτων υλών ή προϊόντων
- Έλεγχος ποιότητας
- Χωροθέτηση εργοστασίου
- Ανανέωση μηχανολογικού εξοπλισμού
- Ανάλυση αξιοπιστίας
- Εξισορρόπηση γραμμής παραγωγής

1.3 Βιβλιογραφική ανασκόπηση

Μεγάλο πλήθος εργασιών και ερευνών που σχετίζονται με το θέμα της βελτιστοποίησης του προγραμματισμού παραγωγής τροφίμων μπορούν να συναντηθούν στη βιβλιογραφία της επιχειρησιακής έρευνας. Παρόλα αυτά, οι προσεγγίσεις του θέματος στην κοινότητα (PSE) διαφέρουν σημαντικά λόγω της πολυπλοκότητας της παραγωγικής διαδικασίας, του μεγάλου πλήθους τυποποιήσεων και τελικών προϊόντων, καθώς και πληθώρας ποιοτικών περιορισμών, τόσο κατά το στάδιο της συσκευασίας, όσο και της αποθήκευσης. Τέλος, σημαντικός παράγοντας στην ανάπτυξη μαθηματικού μοντέλου είναι η αποτύπωση του χρόνου, με τις περισσότερες προσεγγίσεις να έχουν γίνει με μοντέλα συνεχούς χρόνου, όπως οι Doganis P., Sarimvesis H. (2005)[1], τα οποία στερούνται ευελιξίας στα πλαίσια της παύσης και της εναλλαγής προϊόντων, κατά το στάδιο της συσκευασίας, και έτσι καθίστανται ασύμφορα στην εφαρμογή τους σε πραγματικό χρόνο σε περίπτωση οποιασδήποτε βλάβης-απόκλισης.

Οι Lutke M., Gunther H. O., Van P., Beek, Grunow M., Seiler T.(2005) [2] παρουσίασαν στη συνεργασία τους τρεις μορφοποιήσεις μεικτού ακέραιου γραμμικού προγραμματισμού (MILP) με συνδυασμό συνεχούς και διακριτούς αποτύπωσης χρόνου για τον προγραμματισμό της συσκευασίας αναδευμένης (stirred) γιαούρτης. Οι συγγραφείς συνυπολόγισαν πλειάδα παραμέτρων, όπως τη διάρκεια ζωής των προϊόντων καθώς και τους περιορισμούς κατά το στάδιο της τυποποίησης. Η προσέγγιση τους όμως δεν εμπεριείχε τον παράγοντα του κόστους από την εναλλαγή των προϊόντων, καθιστώντας έτσι τη λύση τους κατάλληλη πρωτίστως για προβλήματα σχεδιασμού της παραγωγής (planning) και δευτερευόντως για θέματα προγραμματισμού (scheduling), όπου το κόστος της αλληλουχίας των προϊόντων είναι ιδιαίτερα κρίσιμο.

Στην προσέγγισή του προβλήματος από τους Kopanos G, Puigjaner L., Georgiadis M. (2009) [3] περιλαμβάνεται μια πιο πλήρης αποτύπωση του κόστους παραγωγής, καθώς συνυπολογίζεται το κόστος της τυποποίησης και προετοιμασίας του μείγματος. Στο μοντέλο συνεχούς χρόνου που παρουσίασαν, γίνεται ομαδοποίηση των προϊόντων κατά συνταγές με διαφορετικό χρόνο και κόστος διαχείρισης, ενώ προβλέπεται και χρόνος προετοιμασίας-πλυσίματος των συσκευαστικών μηχανών. Παρόλα αυτά, η θεώρηση τους πως η εκπόνηση αυτών των πλυσιμάτων τελείται αυστηρά κατά το τέλος της παραγωγικής ημέρας, στερεί στο μοντέλο την πρακτική εφαρμογή σε περιπτώσεις αδιάλειπτης παραγωγής με συνεχής-κυλιόμενες βάρδιες.

Στην παρούσα εργασία προτείνεται ένα νέο βασικό μοντέλο μεικτού ακέραιου γραμμικού προγραμματισμού (MILP) διακριτού χρόνου με αυξημένη ευελιξία στις εναλλαγές προϊόντων (στα πλαίσια της εφικτότητας καθώς το κόστος συνυπολογίζεται) με σκοπό την πλήρη πρακτική εφαρμογή του μοντέλου και αναπροσαρμογή του προγραμματισμού σε πραγματικό χρόνο. Το μοντέλο βελτιστοποιεί το χρονοπρογραμματισμό της παραγωγής γιαουρτιού σε πολλαπλές γραμμές συσκευασίας, που σε μερικές περιπτώσεις έχουν τη δυνατότητα να λειτουργήσουν εφεδρικά (με διαφορετική ταχύτητα και κόστος παραγωγής). Επιπροσθέτως, έχουν συνυπολογιστεί οι απαραίτητοι χρόνοι πλυσίματος-προετοιμασίας των γραμμών συσκευασίας για λόγους υγιεινής. Το κόστος αποθήκευσης θεωρείται αμελητέο κυρίως λόγω της πολιτικής της εταιρίας, που διαθέτει πολλαπλές εγκαταστάσεις διανομής και αποθήκευσης, ενώ και η μικρή διάρκεια ζωής του προϊόντος επιτάσσει το μοντέλο της just-in-time παραγωγής.

Τέλος, αξίζει να σημειωθεί πως έχει τεθεί ως αυστηρός περιορισμός, η πλήρης (fulfillment) και έγκαιρη (no tardiness) κάλυψη της ζήτησης, καθιστώντας έτσι το αποτέλεσμα της ελαχιστοποίησης του κόστους παραγωγής και ως βέλτιστη λύση μεγιστοποίησης του κέρδους.

1.4 Οργάνωση διπλωματικής εργασίας

Το υπόλοιπο αυτής της διπλωματικής εργασίας χωρίζεται σε τέσσερις ενότητες που καταλαμβάνουν τα Κεφάλαια 2,3,4 και 5 αντίστοιχα. Συγκεκριμένα:

Στο Κεφάλαιο 2 αναλύουμε την παραγωγική διαδικασία του γιαουρτιού με τις ιδιομορφίες και τους περιορισμούς, που παρουσιάζει.

Στο Κεφάλαιο 3 αναπτύσσουμε το μαθηματικό μοντέλο επίλυσης του προβλήματος του χρονοπρογραμματισμού μιας μηχανής και παρουσιάζουμε τα αριθμητικά αποτελέσματα της εφαρμογής του.

Στο Κεφάλαιο 4 αναπτύσσουμε το μαθηματικό μοντέλο επίλυσης του προβλήματος του χρονοπρογραμματισμού πολλαπλών μηχανών και παρουσιάζουμε τα αριθμητικά αποτελέσματα των εφαρμογών του.

Τα τελικά συμπεράσματα της διπλωματικής εργασίας και κατευθύνσεις για περαιτέρω έρευνα παρουσιάζονται στο Κεφάλαιο 5.

2. ΠΑΡΑΓΩΓΗ ΓΙΑΟΥΡΤΙΟΥ

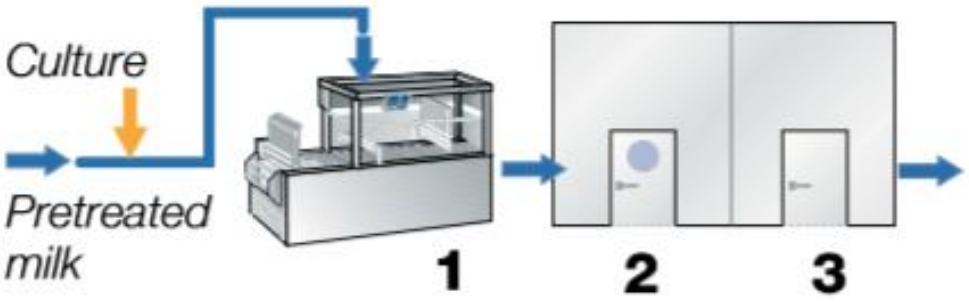
2.1. Ορισμός και κύριοι τύποι γιαουρτιού

Σύμφωνα με τον Codex Alimentarius (FHO/WHO, 1977a) το γιαούρτι ορίζεται ως «πηγμένο γαλακτοκομικό προϊόν που παράγεται με γαλακτική ζύμωση του γάλακτος με τη δράση του Lactobacillus bulgaricus και του Streptococcus thermophilus. Οι μικροοργανισμοί αυτοί πρέπει να είναι στο τελικό προϊόν άφθονοι και ζωντανοί (ελάχιστος αριθμός τους στο τελικό προϊόν είναι 10^7 κύτταρα/g.)»

Οι τρείς κύριοι τύποι γιαουρτιού είναι το συμπαγές ή στερεάς μορφής γιαούρτι (set), το αναδευμένο γιαούρτι (stirred) και το συμπυκνωμένο-στραγγιστό γιαούρτι (strained).

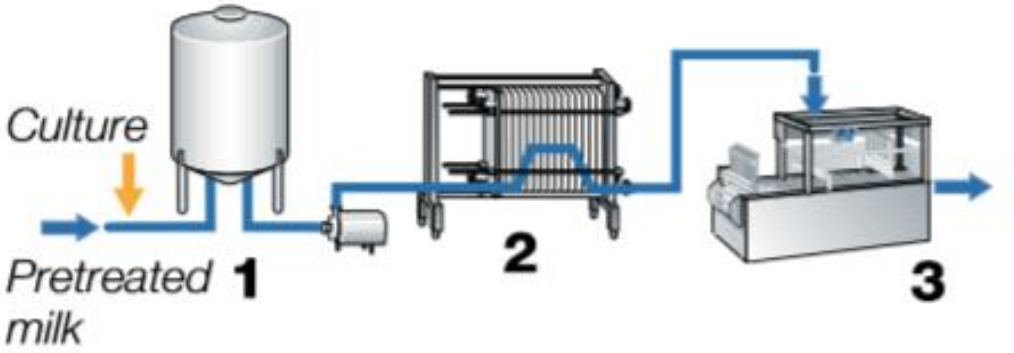
Οι βασικές διαφορές των τριών τύπου γιαουρτιού είναι:

- Στο γιαούρτι στερεάς μορφής, το γάλα συσκευάζεται στους επιθυμητούς περιέκτες και εκεί γίνεται το πήξιμο.



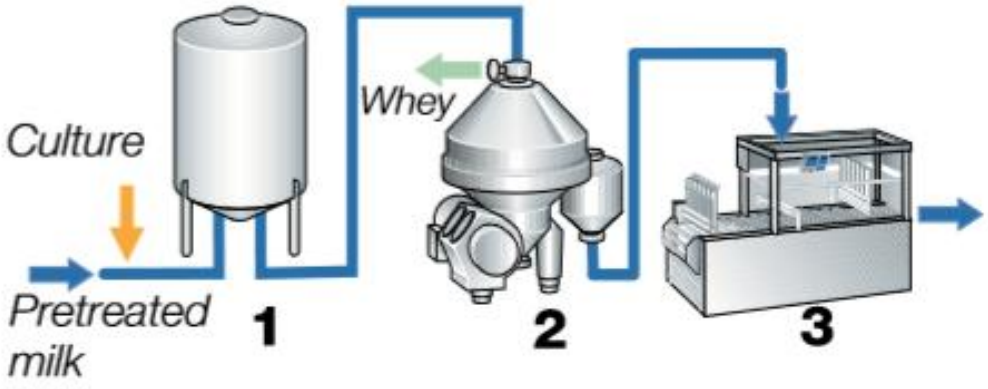
Εικόνα 1: Ροή διεργασιών γιαουρτιού στερεάς μορφής (set)

- Στο αναδευμένο και το στραγγιστό γιαούρτι, το γάλα πήζει σε δεξαμενές, όπου υπόκειται σε ανάδευση και συσκευάζεται σε μεταγενέστερο χρόνο.



Εικόνα 2: Ροή διεργασιών αναδευμένου γιαουρτιού (stirred)

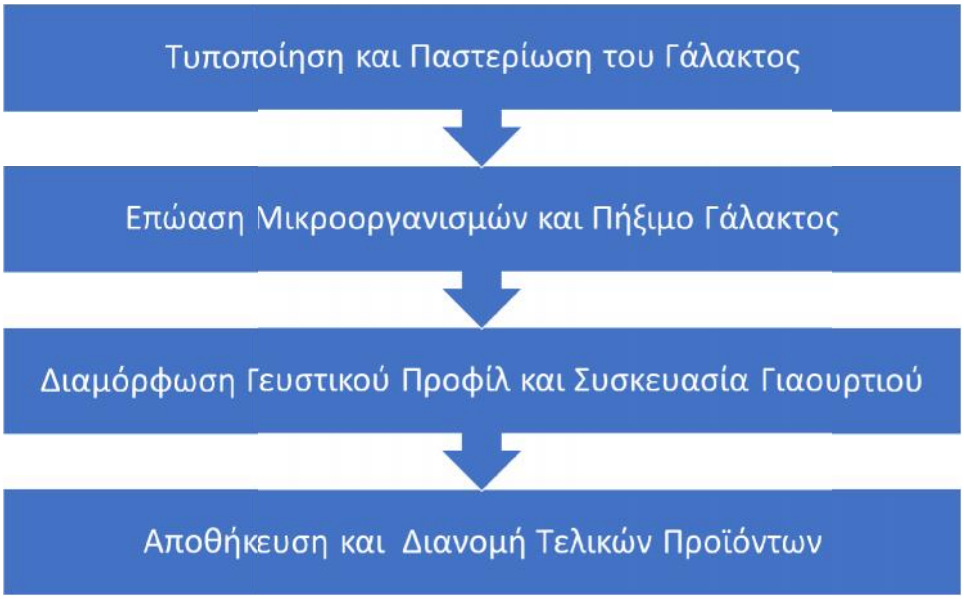
- Στο στραγγιστό γιαούρτι το πήγμα υπόκειται σε αποστράγγιση μέρους του νερού του με τα διαλυόμενα σε αυτό συστατικά, είτε με τον παραδοσιακό τρόπο (υφασμάτινοι σάκοι) είτε με σύγχρονη τεχνολογία (φυγοκέντριση, υπερδιήθηση).
-



Εικόνα 3: Ροή διεργασιών στραγγιστού γιαουρτιού (strained)

2.2. Βασικές διεργασίες παραγωγής γιαουρτιού

Η παραγωγική διαδικασία του γιαουρτιού περιλαμβάνει τέσσερις κύριες διεργασίες (ανεξαρτήτως σειράς εκπόνησης) που αξίζει να αναλυθούν:



Εικόνα 4: Ροή κύριων διεργασιών παραγωγής γιαουρτιού.

1. Τυποποίηση Γάλακτος και Παστερίωση Γάλακτος. Το γάλα συλλέγεται από τους παραγωγούς σε ημερήσια βάση και μεταφέρεται στις εγκαταστάσεις επεξεργασίας του. Εκεί το νωπό γάλα κατά την παραλαβή του ψύχεται και αποθηκεύεται σε δεξαμενές, όπου θα παραμείνει το πολύ για δύο ημέρες. Αρχικά, το νωπό γάλα πρέπει να τυποποιηθεί στα επιθυμητά χαρακτηριστικά του τελικού προϊόντος καθώς η ποιότητα του παραληφθέντος γάλακτος παρουσιάζει διακυμάνσεις. Έπειτα, το τυποποιημένο γάλα υπόκειται σε ομογενοποίηση, όπου τα μεγάλα λιποσφαιρίδια του γάλακτος διασπώνται σε μικρότερου μεγέθους, κάτι που συμβάλλει στην αύξηση του ιξώδους του γάλακτος, στο λευκότερο χρώμα καθώς και στην ομοιομορφία και καλύτερη γεύση του τελικού προϊόντος. Τέλος, το γάλα υπόκειται σε θερμική επεξεργασία (θέρμανση και παραμονή για μικρό διάστημα), ώστε να εξαλειφθούν οι ανεπιθύμητοι και παθογόνοι μικροοργανισμοί και μεταφέρεται στις δεξαμενές επώασης. Οι διαδικασίες αυτές είναι συνήθως πλήρως αυτοματοποιημένες και δεν εμφανίζουν αποκλίσεις.

2. Επώαση Μικροοργανισμών και Πήξιμο Γάλακτος. Όπως περιγράψαμε προηγουμένως η διαδικασία της επώασης μπορεί να προηγείται της συσκευασίας (stirred, strained) είτε να έπεται αυτής (set). Σε κάθε περίπτωση για να πήξει το γάλα κρίνεται απαραίτητη η προσθήκη καλλιέργειας εκκίνησης (Lactobacillus bulgaricus, Streptococcus thermophilus) καθώς και η άσκηση θερμότητας. Ανάλογα με την καλλιέργεια επώασης, τον τύπο γιαουρτιού και τη θερμοκρασία επώασης ο χρόνος που χρειάζεται το γάλα για να πήξει μπορεί να διαφέρει σημαντικά.

3. Διαμόρφωση Γευστικού Προφίλ και Συσκευασία Γιαουρτιού. Τα τελικά προϊόντα γιαουρτιού ακόμα και του ιδίου τύπου διαφέρουν σημαντικά μεταξύ τους στα πλαίσια της γεύσης και των υλικών συσκευασίας. Το γευστικό προφίλ του τελικού προϊόντος μπορεί να διαφοροποιηθεί με προσθήκη φρούτου ή άλλου συστατικού κατά το στάδιο της συσκευασίας του. Οι γραμμές συσκευασίας γιαουρτιού εξαρτώνται κυρίως από το επιθυμητό βάρος του τελικού προϊόντος και μπορεί να διαφέρουν σημαντικά στην ταχύτητα τους. Έτσι λοιπόν, μεγάλη ποικιλία υπάρχει τόσο στο μέγεθος των περιεκτών (150gr-5kg) όσο και στα υλικά από τα οποία είναι κατασκευασμένοι. Ανεξαρτήτως υλικού οι περιέκτες υπόκεινται σε μία σειρά αποστειρώσεων πριν έρθουν σε επαφή με το προϊόν. Η συσκευασία σε πλαίσια ασηπτικότητας και τηρώντας υψηλά επίπεδα υγιεινής επαυξάνει τη διάρκεια ζωής του τελικού προϊόντος. Αυτό επιτάσσει μια σειρά πλυσιμάτων και αποστειρώσεων του εξοπλισμού, που μπορεί να διαρκέσει αρκετές ώρες. Οι διαδικασίας αυτές είναι αυτοματοποιημένες και εξασφαλίζουν την απομάκρυνση υπολειμμάτων προϊόντος (product loss) καθώς και παθογόνων μικροοργανισμών.

4. Αποθήκευση και Διανομή Τελικών Προϊόντων. Μετά τη συσκευασία (και το στάδιο της επώασης αν πρόκειται για γιαούρτι τύπου set) το τελικό προϊόν πρέπει να ψυχθεί σε θερμοκρασία κάτω των 10°C, ώστε να αδρανοποιηθεί πλήρως η καλλιέργεια. Η αποθήκευση του προϊόντος σε χαμηλή θερμοκρασία για περίοδο δύο έως πέντε ημερών είναι απαραίτητη για τη σταθεροποίηση την ποιότητας του τελικού προϊόντος (δομή, γεύση). Τέλος κατά το στάδιο αυτό γίνεται ο ποιοτικός έλεγχος του προϊόντος πριν διοχετευτεί στην αγορά μέσω των καναλιών διανομής (supply chain).

2.3. Μελέτη γραμμής παραγωγής γιαουρτιού

2.3.1 Κατηγοριοποίηση των προϊόντων

Η περίπτωση της γαλακτοβιομηχανίας που μελετάμε περιλαμβάνει τη συσκευασία προϊόντων γιαουρτιού σε πλαστικούς περιέκτες από πέντε γραμμές συσκευασίας. Αυτές οι γραμμές συσκευασίας διαχωρίζονται σύμφωνα τη χωρητικότητα των περιεκτών, που μπορούν να διαχειριστούν. Συγκεκριμένα, η γαλακτοβιομηχανία διαθέτει:

- Δύο γραμμές συσκευασίας πλαστικών κυπέλλων χωρητικότητας 150-450 γραμμαρίων.
- Μία γραμμή συσκευασίας πλαστικών κουβάδων 1000 γραμμαρίων.
- Μία γραμμή συσκευασίας πλαστικών κουβάδων 5000 γραμμαρίων.
- Μία γραμμή συσκευασίας πλαστικών κυπέλλων 135 γραμμαρίων.

Επιπροσθέτως, οι γραμμές συσκευασίας διαθέτουν ειδικές διαμορφώσεις (format) για τη διαχείριση εξεζητημένων προϊόντων, όπως είναι τα διμερή κύπελλα (split cups) καθώς και τις απαραίτητες απολήξεις των γεμιστικών μηχανών (Filling machines) για τη διαχείριση του γιαουρτιού (στο στάδιο της συσκευασίας) τόσο σε ρευστή μορφή γάλακτος (set) όσο κι σε ημιστερεή-πηγμένη μορφή (strained, stirred).Συμπερασματικά, η γαλακτοβιομηχανία δραστηριοποιείται στην παραγωγή τριών τύπων γιαουρτιού ευρωπαϊκού τύπου (set), αναδευμένου τύπου (stirred) και συμπυκνωμένου-στραγγιστού τύπου (strained), με τα αντίστοιχα μείγματα να προετοιμάζονται σε κατάλληλες για τον κάθε τύπο δεξαμενές αποθήκευσης.

Τα τελικά προϊόντα απευθύνονται τόσο για την ελληνική αγορά όσο και για πληθώρα αγορών του εξωτερικού, επιτάσσοντας λοιπόν τη συμμόρφωση των διαδικασιών και των σημάνσεων των προϊόντων στα νομικά πλαίσια της εκάστοτε αγοράς (γλώσσα αναγραφής, στοιχεία παραγωγής, ετικέτες).

Τα τελικά προϊόντα διαφοροποιούνται επίσης στα εξής πλαίσια:

1. **Είδος του γάλακτος**: Αγελαδινό, κατσικίσιο και πρόβειο.
2. **Γευστικό προφίλ**: Προσθήκη φρουτοπαρασκευάσματος, σύσταση λιπαρών.
3. **Προσθήκη συστατικών**: Λακτάση, βιταμίνες.
4. **Υλικά συσκευασίας**: Μακέτες υλικών, εκτύπωση ημερομηνίας.

Συνοψίζοντας, εξαιτίας όλων αυτών των διαφοροποιήσεων τη συσκευασία του γιαουρτιού διέπουν σημαντικοί περιορισμοί αλληλουχίας προϊόντων, ώστε να αποφευχθεί η μετανάστευση χρώματος, η αλλοίωση γεύσης αλλά και ανεπιθύμητη παρουσία αλλεργιογόνων σε μη χαρακτηρισμένα καταλλήλως προϊόντα. Το σύνολο αυτών των κανόνων θα αναλυθεί και θα αποτυπωθεί μαθηματικά στο επόμενο κεφάλαιο (περιορισμοί μοντέλου μορφοποίησης).

Τέλος, τα μείγματα γιαουρτιού και κατ’ επέκταση τα τελικά προϊόντα διαχωρίζονται αναλόγως με την περιεκτικότητά τους σε λιπαρά. Συγκεκριμένα, η σύσταση σε λιπαρά μπορεί ναι είναι 0%, 2%, 4%, 5%, 7% και 10% με την προσθήκη της απαραίτητης ποσότητας κρέμας να μπορεί να γίνει σε διάφορα στάδια της παραγωγικής διαδικασίας.

2.3.2 Το σύστημα παραγωγής του εργοστασίου

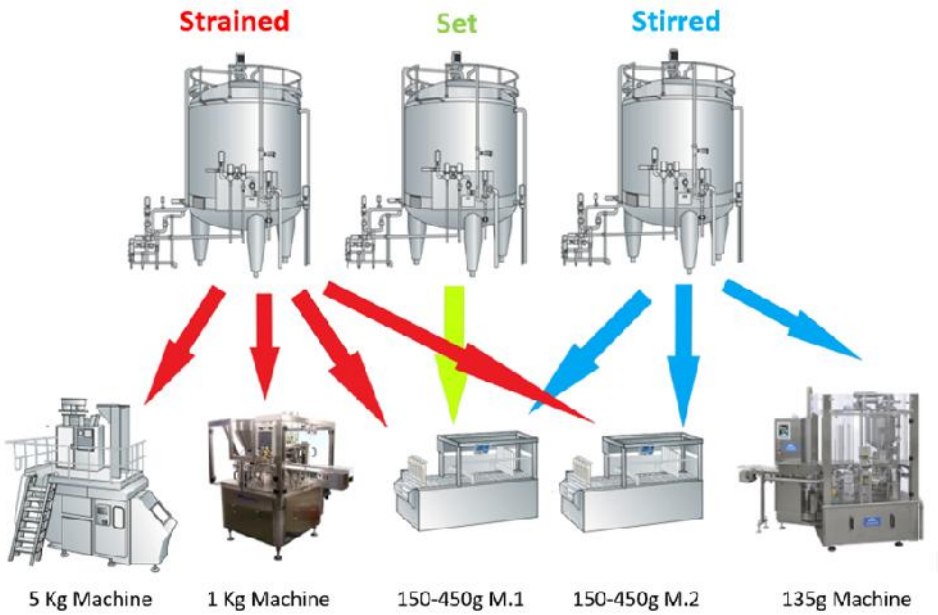
Όπως αναλύθηκε η παραγωγική διαδικασία του γιαουρτιού συνοψίζεται σε τέσσερα κύρια στάδια:

1. Τυποποίηση Γάλακτος και Παστερίωση Γάλακτος.
2. Επώαση Μικροοργανισμών και Πήξιμο Γάλακτος.
3. Διαμόρφωση Γευστικού Προφίλ και Συσκευασία Γιαουρτιού.
4. Αποθήκευση και Διανομή Τελικών Προϊόντων.

Γίνεται εύκολα αντιληπτό πως το στάδιο της συσκευασίας-πλήρωσης των προϊόντων αποτελεί το κρισιμότερο σημείο της παραγωγικής διαδικασίας του γιαουρτιού καθώς κατά τη διάρκεια του το προϊόν έρχεται σε επαφή με το περιβάλλον

(γεμιστικές μηχανές), ενώ τέλος διαμορφώνεται το σύνολο των ποιοτικών του χαρακτηριστικών.

Οι κατασκευαστικοί οίκοι μηχανών παραγωγής και συσκευασίας τροφίμων έχουν αναπτύξει εξελιγμένα συστήματα, που επιτρέπουν τη διαχείριση πληθώρας διαφορετικών συσκευασιών σε ασηπτικό περιβάλλον. Όπως προαναφέραμε, το σύστημα της βιομηχανίας που εξετάζουμε περιλαμβάνει την παραγωγή τριών τύπων γιαουρτιού, η τυποποίηση και επώαση των οποίων λαμβάνει χώρα σε τρεις ενδιάμεσες δεξαμενές-σιλό και συσκευάζονται σε ένα σύνολο πέντε μηχανών-γραμμών κλιμακούμενης χωρητικότητας. Οι δεξαμενές επώασης συνδέονται με τις γεμιστικές μηχανές, μέσω κλειστού συστήματος σωληνώσεων εξασφαλίζοντας μεγάλες παραγωγικές δυνατότητες στα πλαίσια της διαφοροποίησης. Το υφιστάμενο κωδικολόγιο περιλαμβάνει τη συσκευασία γιαουρτιού ευρωπαϊκού τύπου (set) στη μία μηχανή πλαστικών κυπέλλων 150-450 γρ., τη συσκευασία γιαουρτιού αναδευμένου τύπου (stirred) στις δύο μηχανές πλαστικών κυπέλλων 150-450 γρ. και στη μηχανή 135 γρ. ενώ τέλος το στραγγιστό γιαούρτι συσκευάζεται σε όλες τις μηχανές εκτός αυτής των 135 γρ. Η συγκεκριμένη σύνθεση παρουσιάζεται στο παρακάτω σχήμα:



Εικόνα 5: Σχηματική απεικόνιση συνδεσιμότητας εξοπλισμού

Τα τελικά προϊόντα μετά το στάδιο της συσκευασίας μεταφέρονται σε εγκαταστάσεις αποθήκευσης-ψύξης, στις οποίες παραμένουν για συγκεκριμένο

χρονικό διάστημα πριν διοχετευτούν στην αγορά. Ο συνολικός χρόνος μεταφοράς και αποθήκευσης είναι κατά ελάχιστο τέσσερις ημέρες. Αυτό το διάστημα είναι απαραίτητο για τον ποιοτικό έλεγχο και την σταθεροποίηση της οξύτητας των τελικών προϊόντων.

Στο τμήμα γιαούρτης λόγω περιορισμών χώρου και εξοπλισμού υποστηρίζεται κατά μέγιστο η ταυτόχρονη λειτουργία μέχρι δύο συσκευαστικών μηχανών, για το χειρισμό των οποίων απαιτούνται τέσσερις εργαζόμενοι. Ως εκ τούτου κάθε βάρδια (στα πλαίσια της συσκευασίας) αποτελείται από οκτώ εργαζόμενους, ενώ τα χρονικά διαστήματα για τις βάρδιες είναι τα εξής:

• Για την πρωινή βάρδια [7:00-15:00],

• Για την απογευματινή βάρδια [15:00-23:00],

• Για την νυχτερινή [23:00-7:00].

Το εργατικό κόστος, δηλαδή η μισθοδοσία των εργατών είναι για την πρωινή και την απογευματινή βάρδια το ίδιο. Η νυχτερινή βάρδια πληρώνεται επιπλέον 25% του κανονικού μισθού. Σε περιόδους που πρέπει να υπάρχει παραγωγή Κυριακές και ημέρες αργιών ο μισθός είναι συν 75% του κανονικού. Θα πρέπει να τονίσουμε ότι το εργοστάσιο αποφεύγει να βάζει εργαζόμενους στην 3η βάρδια και τις αργίες.

Ο προγραμματισμού της παραγωγής γίνεται σε ορίζοντα μίας εβδομάδας εκκινώντας κάθε Δευτέρα. Η καταχώρηση της ζήτησης από τις εγκαταστάσεις αποθήκευσης γίνεται με μορφή εβδομαδιαίας πρόβλεψης μετά από ανάλυση διαθεσιμότητας αποθεμάτων με καταληκτική ημερομηνία παράδοσης την επόμενη Κυριακή. Γίνεται λοιπόν αντιληπτό πως κατά τον προγραμματισμό της παραγωγής δε συνυπολογίζεται το αρχικό απόθεμα (starting inventory) από τον υπεύθυνο εκπόνησης του προγράμματος (scheduler). Επίσης τα τελικά προϊόντα μεταφέρονται στις εγκαταστάσεις αποθήκευσης μετά την παραγωγή τους και έτσι το εργοστάσιο παραγωγής δεν επιβαρύνεται με το αντίστοιχο κόστος. Στα μοντέλα που αναπτύχθηκαν υπάρχει διακριτή αναπαράσταση του χρόνου (discrete time representation) και χωρίζεται το διάστημα της μίας εβδομάδας σε ώρες, επομένως σε 168 περιόδους.

Το σύνολο των κωδικών έχει ομαδοποιηθεί σε είκοσι υποσύνολα με προϊόντα ίδιων χαρακτηριστικών, τα οποία περιλαμβάνουν όλες τις διαφοροποιήσεις των παραγωγικών παραμέτρων. Έτσι η παραγωγή εκτελείται σε παρτίδες (batch production) ικανού όγκου. Τα υποσύνολα αυτά διέπουν σημαντικοί περιορισμοί αλληλουχίας, στα πλαίσια της εφικτότητας εναλλαγής διαμορφώσεων των μηχανών (format) αλλά και στη διασφάλιση της ποιότητας των προϊόντων. Οι περιορισμοί αυτοί μπορεί να απαγορεύουν την παραγωγή μίας ή όλων των ομάδων προϊόντων πριν και μετά τη συσκευασία κάποιας ομάδας. Αναλυτικά:

- Free lact: Δε μπορεί να έπεται ομάδας προϊόντων φέρουσας λακτόζης στην ίδια μηχανή.
- Split Cup: Δε μπορεί να έπεται ούτε να προηγείται της συσκευασίας ομάδας προϊόντος στην ίδια μηχανή, λόγω format.
- Στραγγιστό Mix: Δε μπορεί να προηγείται ομάδας προϊόντος στην ίδια μηχανή, λόγω μετανάστευσης χρώματος.
- Κεφίρ: Δε μπορεί να προηγείται ομάδας προϊόντος στην ίδια μηχανή, λόγω μετανάστευσης χρώματος.
- Ομάδες τύπου set: Δε μπορούν να έπονται ούτε να προηγούνται της συσκευασίας ομάδας προϊόντος άλλου τύπου στην ίδια μηχανή, λόγω ιδιαιτερότητας γεμιστικών απολήξεων.
- Παιδικό μπισκότο: Δε μπορεί να προηγείται ομάδας προϊόντος στην ίδια μηχανή, λόγω παρουσίας γλουτένης.

ΠΡΙΝ	ΜΕΤΑ	Ομάδες Προϊόντων
ΟΧΙ		FREE LACT 150g
		ΣΤΡΑΓΓΙΣΤΟ 10% 5kg
		ΣΤΡΑΓΓΙΣΤΟ 10% 1kg
		ΣΤΡΑΓΓΙΣΤΟ 10% 200g
		ΣΤΡΑΓΓΙΣΤΟ 2% 1kg
		ΣΤΡΑΓΓΙΣΤΟ 2% 200g
ΟΧΙ	ΟΧΙ	ΣΤΡΑΓΓΙΣΤΟ 2% split cup
		ΣΤΡΑΓΓΙΣΤΟ 0% 1kg
		ΣΤΡΑΓΓΙΣΤΟ 0% 200g
		ΣΤΡΑΓΓΙΣΤΟ 0% UNDER 150g
	ΟΧΙ	ΣΤΡΑΓΓΙΣΤΟ 0% MIX 150g
		ΠΝΟΗ 170g
	ΟΧΙ	ΚΕΦΙΡ 150g
ΟΧΙ	ΟΧΙ	ΚΑΤΣΙΚΙΣΙΟ 200g
		ΠΡΟΒΕΙΟ 200g
		ΑΓΕΛΑΔΟΣ 2% 200g
		ΒΙΟΛΟΓΙΚΟ 200g
		ΠΑΙΔΙΚΟ ΜΠΑΝΑΝΑ
		ΠΑΙΔΙΚΟ ΦΡΑΟΥΛΑ
	ΟΧΙ	ΠΑΙΔΙΚΟ ΜΠΙΣΚΟΤΟ

Πίνακας 2: Παρουσία ομάδων προϊόντων και περιορισμών αλληλουχίας

Τέλος, λόγω ασηπτικότητας της γραμμής παραγωγής επιτρέπεται η συνεχής λειτουργία (χωρίς ενδιάμεσο πλύσιμο) των γεμιστικών μηχανών για περίοδο 22 ωρών στα πλαίσια τήρησης των υγειονομικών προτύπων. Μετά το πέρας του συγκεκριμένου διαστήματος ο εξοπλισμός πρέπει να καθαριστεί με τη βοήθεια αυτοματοποιημένου συστήματος CIP (Cleaning In Place), ο χρόνος διάρκειας του οποίου είναι δύο ώρες.

2.3.3 Στόχοι της επιχείρησης

Η γαλακτοβιομηχανία έχει θέσει ως στρατηγική προτεραιότητα την πλήρη ικανοποίηση των πελατών-ζήτησης παρέχοντας μεγάλο φάσμα προϊόντων υψηλής

ποιότητας. Τα κριτήρια λοιπόν με τα οποία λαμβάνοντα οι αποφάσεις της διοίκησης παραγωγής είναι τα εξής:

- Κάλυψη της ζήτησης εντός του ορίζοντα φόρτωσης. Στόχος του προγραμματισμού της παραγωγής είναι η έγκαιρη κάλυψη (no tardiness) του συνόλου της ζήτησης καθώς, λόγω της μικρής διάρκειας ζωής των προϊόντων, καθυστέρηση της παράδοσης επιφέρει αποτυχία πώλησης (no backordering).
- Αποφυγή υπερωριών. Η εταιρία έχει θέσει ως πολιτική την τήρηση του νόμιμου ωραρίου εργασίας με εναλλαγή κυλιόμενων βαρδιών, ώστε να εξασφαλίσει την επιτυχή εκπόνηση των παραγωγικών διαδικασιών τηρώντας υψηλά επίπεδα ποιότητας (καθώς πρόκειται για διεργασίες με μεγάλο υγειονομικά ενδιαφέρον) και να μειώσει μακροπρόθεσμα το λειτουργικό κόστος.
- Ομαδοποίηση των προϊόντων ώστε οι παραγωγές να είναι μαζικές (batch production) και να ελαττωθεί το κόστος εναλλαγής προϊόντων. Τα προϊόντα που έχουν ίδιες παραμέτρους παραγωγής (τύπος γιαουρτιού, τυποποίηση λιπαρών, γευστικό προφίλ) έχουν χωριστεί σε ευρύτερες ομάδες (στις οποίες τηρούνται περιορισμοί αλληλουχίας), ώστε να μειωθεί το κόστος και το χρόνος για την προετοιμασία των μειγμάτων και των μηχανών.
- Προσπάθεια εκμετάλλευσης της μέγιστης δυναμικότητας της μηχανής εντός προτύπων υγιεινής. Στα πλαίσια της ασηπτικής συσκευασίας γίνεται ομαδοποίηση των παρτίδων παραγωγής, ώστε να ελαχιστοποιηθεί ο χρόνος και το κόστος που απαιτείται για τον καθαρισμό και την προετοιμασία της μηχανής.

Εικόνα 6: Σχηματική απεικόνιση στρατηγικών προτεραιοτήτων

3. ΜΟΡΦΟΠΟΙΗΣΗ ΤΟΥ ΜΑΘΗΜΑΤΙΚΟΥ ΜΟΝΤΕΛΟΥ ΜΟΝΗΣ ΓΡΑΜΜΗΣ (SINGLE MACHINE)

3.1. Δείκτες

Δείκτες-Σύνολα	
t	Χρονικές περίοδοι (1,2… T).
i	Τελικά προϊόντα (1,2… P).

Πίνακας 3: Δείκτες μαθηματικού μοντέλου (single machine)

3.2. Παράμετροι

Παράμετροι	
D_i	Συνολική ζήτηση για το προϊόν i.
P_i	Ποσότητα παραγωγής του προϊόντος i στη μηχανή για μία χρονική περίοδο/ελάχιστη συνολική ποσότητα παραγωγής.
Cl	Κόστος εργατικών απασχόλησης 4 ατόμων για μία χρονική περίοδο πρωινή και απογευματινή βάρδια.
Cln	Κόστος εργατικών απασχόλησης 4 ατόμων για μία χρονική περίοδο βραδινή βάρδια.
Clh	Κόστος εργατικών απασχόλησης 4 ατόμων για μία χρονική περίοδο σε αργία.
Cs_i	Κόστος προετοιμασίας για τη συσκευασία του προϊόντος ¡.
Cc	Κόστος πλυσίματος της μηχανής σε μία χρονική περίοδο.

Πίνακας 4: Παράμετροι μαθηματικού μοντέλου (single machine)

3.3. Μεταβλητές απόφασης

Μεταβλητές απόφασης	
$X_{t,i}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν το προϊόν i παράγεται στη μηχανή τη χρονική περίοδο t.
$Y_{t,i}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή ετοιμάζεται για το προϊόν i τη χρονική περίοδο t.
Z_t	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή πλένεται τη χρονική περίοδο t.
c_t	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή ξεκινάει να πλένεται τη χρονική περίοδο t.
K_t	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή είναι σε κατάσταση αναμονής τη χρονική περίοδο t.

Πίνακας 5: Μεταβλητές απόφασης μαθηματικού μοντέλου (single machine)

3.4. Αντικειμενική συνάρτηση

$$M \sum_{t=1}^{1}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C + \sum_{t=1}^{2}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C +$$

$$+\sum_{t=2}^{4}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C + \sum_{t=4}^{4}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C +$$

$$+\sum_{t=4}^{6}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C + \sum_{t=6}^{7}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C +$$

$$+\sum_{t=7}^{8}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C + \sum_{t=8}^{9}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C +$$

$$+\sum_{t=9}^{1}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C + \sum_{t=1}^{1}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C +$$

$$+\sum_{t=1}^{1}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C\ +\sum_{t=1}^{1}\sum_{i}^{P}\left(X_{t,i}+Y_{t,i}\right)\cdot C\quad +$$

$$+\sum_{t=1}^{1}\sum_{i}^{P}(X_{t,i}+Y_{t,i})\cdot C\ h+\sum_{t=1}^{1}\sum_{i=1}^{P}Y_{t,i}\cdot C_{\ i}+\sum_{t=1}^{1}Z_{t}\cdot C\qquad(1)$$

Η αντικειμενική συνάρτηση αποτελείται από τρείς συνιστώσες, οι οποίες αποτυπώνουν 1) το κόστος των εργατικών, 2) το κόστος εναλλαγής προϊόντων και 3) το κόστος πλυσιμάτων για όλες τις χρονικές περιόδους στον ορίζοντα προγραμματισμού και αναλύονται ως εξής:

1) **<u>Το κόστος εργατικών</u>**: Αποτυπώνει τις εργατοώρες που απαιτούνται για την εκπόνηση του προγράμματος παραγωγής και είναι το γινόμενο του κυμαινόμενου κόστους απασχόλησης μίας ομάδας 4 εργαζομένων για μια χρονική περίοδο με το πλήθος των χρονικών περιόδων για διεργασίες, που απαιτούν ανθρώπινο χειρισμό (Λειτουργία συσκευαστικών μηχανών και προετοιμασία συσκευαστικών μηχανών για αλλαγή προϊόντος). Είναι το άθροισμα 13 επιμέρους όρων διακριτοποιημένων χρονικών περιόδων με το αντίστοιχο κόστος για την εκάστοτε περίοδο.
2) **<u>Το κόστος εναλλαγής προϊόντων</u>**: Αποτυπώνει το κόστος εναλλαγής από προϊόν σε προϊόν στη συσκευαστική μηχανή αναπαριστώντας τις απώλειες πρώτης ύλης για το άδειασμα των δεξαμενών εξισορρόπησης (buffer tanks) της μηχανής, ώστε αυτή να καταστεί έτοιμη για τη συσκευασία του επόμενου προϊόντος δίχως αποκλίσεις στην ποιότητα του (ποιοτικά χαρακτηριστικά, μετανάστευση χρώματος). Προκύπτει λοιπόν από το άθροισμα των γινομένων όλων των εναλλαγών που εκπονούνται με το κόστος της εκάστοτε εναλλαγής.
3) **<u>Το κόστος πλυσιμάτων</u>**: Αποτυπώνει το κόστος του πλυσίματος του εμπλεκόμενου εξοπλισμού καθώς και την ανάλωση των χημικών. Πρόκειται για το μεγαλύτερο τη τάξη κόστος καθώς περιλαμβάνει τις απώλειες πρώτης ύλης σε όλο το φάσμα της γραμμής παραγωγής (δεξαμενές, σωληνώσεις, μηχανές). Προκύπτει λοιπόν από το γινόμενο του πλήθος των χρονικών περιόδων που λαμβάνει χώρα κάποιο πλύσιμο της μηχανής με το κόστος του.

3.5. Περιορισμοί

$$\sum_{i=1}^{P}(X_{t,i} + Y_{t,i}) + K_t + Z_t = 1 \quad \forall t \quad (2)$$

Ο περιορισμός (2) εξασφαλίζει πως ανά πάσα στιγμή η μηχανή θα είναι αυστηρά σε μία και μόνο κατάσταση (παραγωγή, προετοιμασία, αναμονή, πλύσιμο).

$$\sum_{t-1}^{T} P_i \cdot X_{t,i} \geq D_i \forall i \quad (3)$$

Ο περιορισμός (3) εξασφαλίζει πως η ζήτηση κάθε προϊόντος θα καλυφθεί εντός του χρονικού ορίζοντα παραγωγής.

$$\sum_{t-1}^{T} P_i \cdot X_{t,i} \leq D_i + P \quad \forall i \quad (4)$$

Ο περιορισμός (4) εξασφαλίζει πως η παραγωγή του κάθε προϊόντος δε θα ξεπεράσει τη ζήτηση κατά μεγάλο ποσοστό εντός του χρονικού ορίζοντα παραγωγής (καθώς η πλεονάζουσα ποσότητα δε μπορεί να διατεθεί στην αγορά).

$$Y_{t,i} \geq X_{t+1,i} - X_{t,i} \quad t = 1 \ldots . T - 1, \forall i \quad (5)$$

$$X_{t+1,i} - X_{t,i} + 1{,}1\left(1 - Y_{t+1,i}\right) \geq 0{,}1 \quad t = 1 \ldots . T - 1, \forall i \quad (6)$$

Ο συνδυασμός των περιορισμών (5) και (6) εξασφαλίζει πως η μηχανή θα τεθεί σε κατάσταση προετοιμασίας πριν την έναρξη της παραγωγής οποιουδήποτε προϊόντος και θα ολοκληρώσει την προετοιμασίας της με την έναρξη της παραγωγής.

$$Z_{t+1} \geq \sum_{i=1}^{P}(X_{t,i} + Y_{t,i}) - \sum_{i=1}^{P}(X_{t+1,i} + Y_{t+1,i}) \quad t = 1 \ldots . T - 1, \forall i \quad (7)$$

Ο περιορισμός (7) εξασφαλίζει πως η μηχανή θα τεθεί σε κατάσταση πλυσίματος μετά το πέρας κάποιας παραγωγικής δραστηριότητας.

$$c_{t+1} \geq Z_{t+1} - Z_t \quad t = 1 \dots . T - 1 \quad (8)$$

$$Z_{t+1} - Z_t + 1{,}1(1 - c_{t+1}) \geq 0{,}1 \quad t = 1 \dots . T - 1 \quad (9)$$

Ο συνδυασμός των περιορισμών (8) και (9) εξασφαλίζει πως η μεταβλητή έναρξης πλυσίματος θα πάρει τη σωστή τιμή αν η μηχανή τεθεί σε κατάσταση πλυσίματος.

$$Z_t + Z_{t+1} \geq 2 \cdot c_t t = 1 \dots . T - 1 \quad (10)$$

Ο περιορισμός (10) εξασφαλίζει πως η μηχανή θα διατηρηθεί σε κατάσταση πλυσίματος για τις απαραίτητες χρονικές περιόδους.

$$\sum_{\tau=t}^{t+2} \sum_{i}^{P} (X_\tau + Y_\tau) \leq 22 \quad t = 1 \dots . T - 23 \quad (11)$$

Ο περιορισμός (11) εξασφαλίζει πως η μηχανή δε θα ξεπεράσει τις 22 ώρες συνεχούς λειτουργίας (τήρηση επιπέδων υγιεινής).

$$\sum_{t=1}^{T} Y_{t,i} \leq 2 + \mu(D - 22 \cdot P) \forall i (12)$$

Ο περιορισμός (12) εξασφαλίζει πως αν η ζήτηση ενός προϊόντος μπορεί να καλυφθεί με μία και μόνο προετοιμασία της μηχανής η παραγωγή του θα εκτελεστεί με συνεχή τρόπο (ομαδοποίηση παραγωγής-επιτάχυνση βελτιστοποίησης).

<u>Περιορισμοί αλληλουχίας (levelling):</u>

Free lact(k=0)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(X_{t-2,i}) + X_{t,k} \leq 1 \quad t = 3 \ldots . T \qquad (13)$$

Ο περιορισμός (13) εξασφαλίζει πως το προϊόν Free lact δε θα έπεται της παραγωγής κάποιου άλλου προϊόντος.

Split cup (k=3)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(X_{t-2,i}) + X_{t,k} \leq 1 \quad t = 3 \ldots . T \qquad (14)$$

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,i}) + X_{t,k} \leq 1 \quad t = 1 \ldots . T - 1 \qquad (15)$$

Ο συνδυασμός των περιορισμών (14) και (15) εξασφαλίζει πως το προϊόν Split cup δε θα έπεται ούτε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος.

Κεφίρ (k=8)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,i}) + X_{t,k} \leq 1 \quad t = 1 \ldots . T - 1 \qquad (16)$$

Ο περιορισμός (16) εξασφαλίζει πως το προϊόν Κεφίρ δε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος.

Στραγγιστό mix(k=6)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,i}) + X_{t,k} \leq 1 \quad t = 1 \ldots . T - 1 \qquad (17)$$

Ο περιορισμός (17) εξασφαλίζει πως το προϊόν Στραγγιστό Mix δε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος.

<u>Αρχικές και τελικές καταστάσεις:</u>

$$X_{t,i} = 0 \quad t = 0, \forall i \qquad (18)$$

Ο περιορισμός (18) εξασφαλίζει πως κανένα προϊόν δε θα παράγεται κατά την αρχική περίοδο του ορίζοντα προγραμματισμού.

$$X_{T-1,i} + X_{T,i} + Y_{T-1,i} + Y_{T,i} = 0 \quad \forall i \qquad (19)$$

Ο περιορισμός (19) εξασφαλίζει πως καμία παραγωγική δραστηριότητα δε θα λαμβάνει χώρα κατά τις δύο τελικές χρονικές περιόδους του ορίζοντα προγραμματισμού (έτσι ώστε η μηχανή να μπορεί να πλυθεί ώστε να είναι έτοιμη να τεθεί σε κάποια παραγωγική δραστηριότητα στην αρχή του επόμενου ορίζοντα).

3.6. Εφαρμογή μοντέλου σε μία μηχανή για 9 προϊόντα

3.6.1. Υπολογιστικά αποτελέσματα

Το κωδικοποιημένο μοντέλο λύθηκε σε υπολογιστικό περιβάλλον CPLEX_Studio125 με χρήση γλώσσα προγραμματισμού C++ σε ηλεκτρονικό υπολογιστή με χαρακτηριστικά Intel®Core™i5-3330 CPU @ 3.00GHz, RAM 8,00 GB παραθέτοντας τα παρακάτω αποτελέσματα:

Solution Value	19323
Time(sec)	6434(107,2 min)
Gap	48,5%

Πίνακας 6 : Υπολογιστικό αποτέλεσμα μοντέλου (single machine)

Για να κατανοήσουμε σε βάθος το αποτέλεσμα της λύσης και να αντλήσουμε τα απαραίτητα συμπεράσματα οφείλουμε να αναλύσουμε τους κοστολογικούς όρους της αντικειμενικής συνάρτησης, την αλληλουχία καταστάσεων στη μηχανή καθώς και τη σειρά εκτέλεσης της παραγωγής των προϊόντων.

3.6.2. Αξιολόγηση αποτελεσμάτων

Όπως προαναφέραμε η αντικειμενική συνάρτηση αποτελείται από τρεις ομαδοποιημένους κοστολογικούς όρους οι οποίοι αποτυπώνουν 1) Το κόστος εργατικών, 2) το κόστος εναλλαγής προϊόντων και 3) το κόστος πλυσιμάτων.

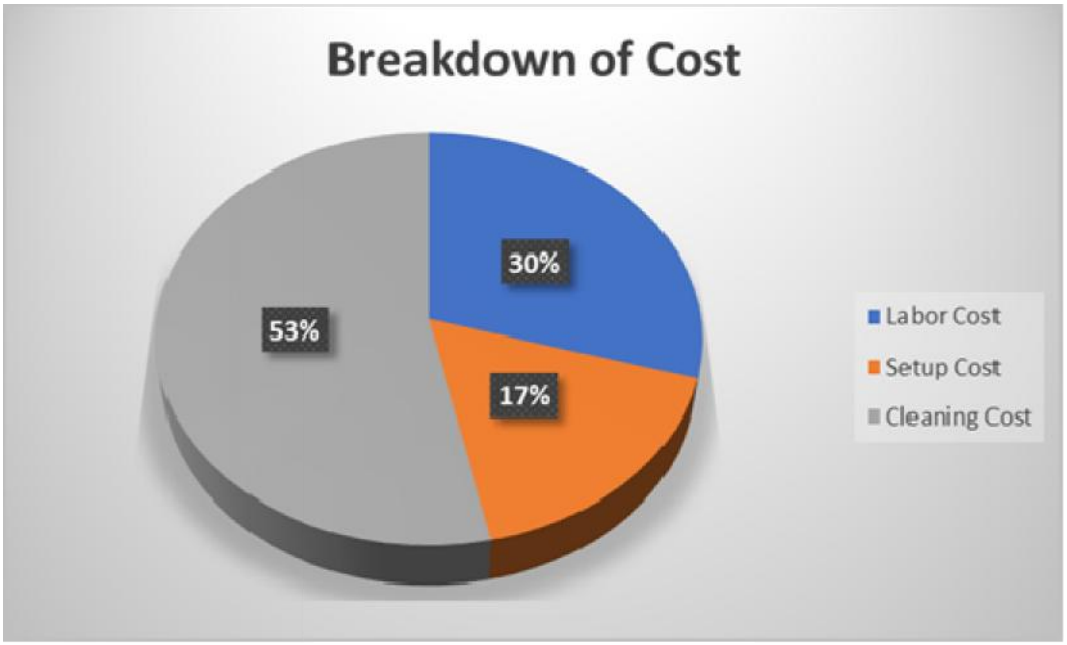
Στη λύση που προέκυψε αυτοί διαμοιράζονται ως εξής:

Costing terms	Cost (€)
Labor Cost	5707
Setup Cost	3302
Cleaning Cost	10314
Total Packaging Cost	19323

Πίνακας 7: Ανάλυση κοστολογικών όρων (single machine)

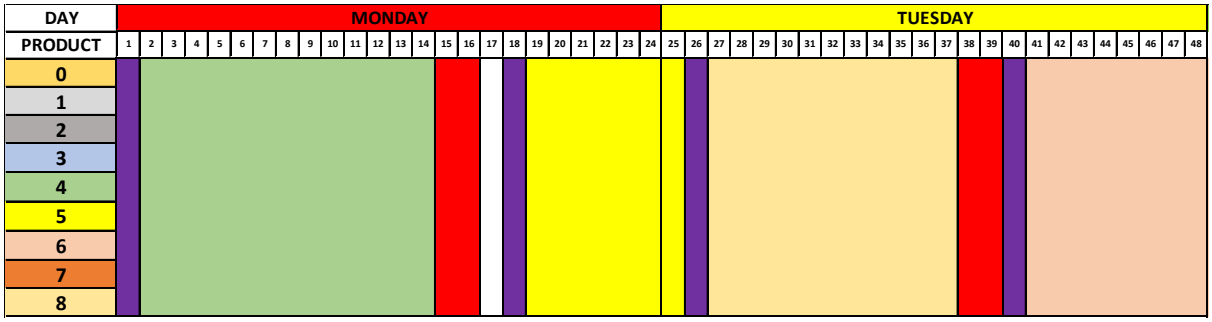
Total Production (kg)	158860,0
Labor Cost (€/ kg)	0,04
Total Pac Cost (€/ kg)	0,12

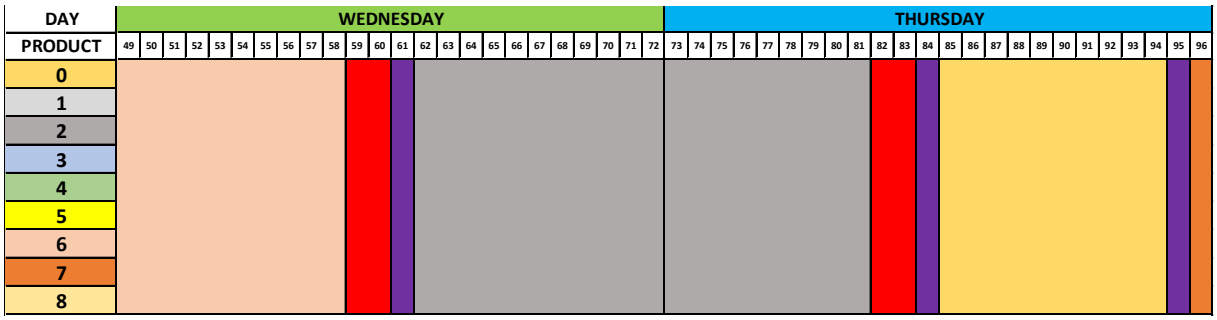
Πίνακας 8: Αναγωγή κοστολογικών όρων επί της παραγωγής (single machine)

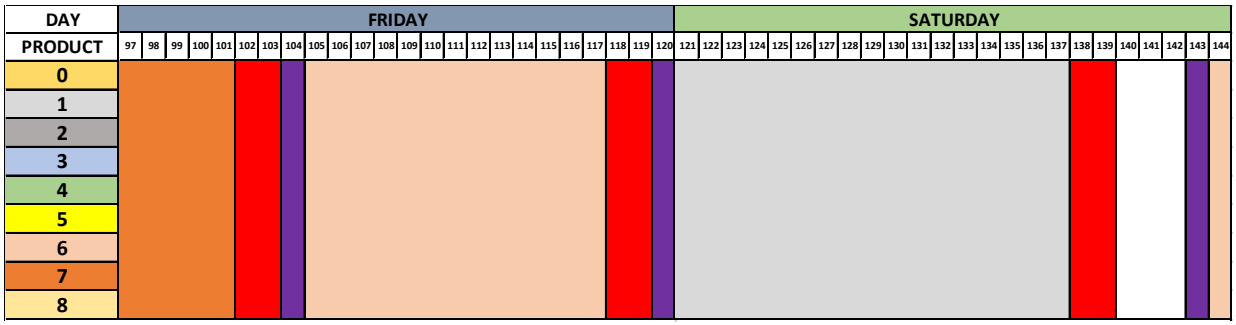


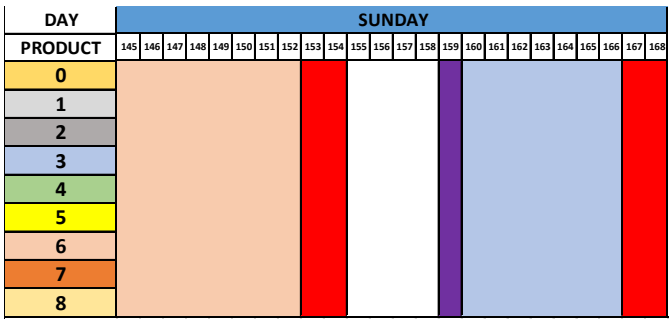
Πίνακας 9: Ποσοστιαία αποτύπωση κοστολογικών όρων (single machine)

Filling machine 150-450 gr.		
Mode	Periods	Utilization rate
X	131	95%
Y	11	
Z	18	
K	8	


DAY
MONDAY
TUESDAY
PRODUCT
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
0
1
2
3
4
5
6
7
8


DAY
WEDNESDAY
THURSDAY
PRODUCT
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
0
1
2
3
4
5
6
7
8


DAY
FRIDAY
SATURDAY
PRODUCT
97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
0
1
2
3
4
5
6
7
8


DAY
SUNDAY
PRODUCT
145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168
0
1
2
3
4
5
6
7
8

4. ΜΟΡΦΟΠΟΙΗΣΗ ΤΟΥ ΔΙΕΥΡΥΜΕΝΟΥ ΜΑΘΗΜΑΤΙΚΟΥ ΜΟΝΤΕΛΟΥ (PARALLEL MACHINES)

4.1. Δείκτες

Δείκτες-Σύνολα	
t	Χρονικές περίοδοι (1,2… T).
m	Μηχανές συσκευασίας (1,2… M).
i	Τελικά προϊόντα (1,2… P).

Πίνακας 11: Δείκτες μαθηματικού μοντέλου (parallel machines)

4.2. Παράμετροι

Παράμετροι	
D_i	Συνολική ζήτηση για το προϊόν i.
$P_{m,i}$	Ποσότητα παραγωγής του προϊόντος i στη μηχανή m για μία χρονική περίοδο.
P_i	Ελάχιστη ποσότητα παραγωγής για το προϊόν i.
Cl	Κόστος εργατικών απασχόλησης μίας βάρδιας 4 ατόμων για μία χρονική περίοδο.
Cs_i	Κόστος προετοιμασίας για τη συσκευασία του προϊόντος i.
Cc_m	Κόστος πλυσίματος της μηχανής m σε μία χρονική περίοδο.

Πίνακας 12: Παράμετροι μαθηματικού μοντέλου (parallel machines)

4.3. Μεταβλητές απόφασης

Μεταβλητές απόφασης	
$X_{t,m,i}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν το

	προϊόν i παράγεται στη μηχανή m τη χρονική περίοδο t.
$Y_{t,m,i}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή m ετοιμάζεται για το προϊόν i τη χρονική περίοδο t.
$Z_{t,m}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή m πλένεται τη χρονική περίοδο t.
$c_{t,m}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή m ξεκινάει να πλένεται τη χρονική περίοδο t.
$K_{t,m}$	Δυαδική μεταβλητή απόφασης, που λαμβάνει την τιμή 1 αν η μηχανή m είναι σε κατάσταση αναμονής τη χρονική περίοδο t.

Πίνακας 13: Μεταβλητές απόφασης μαθηματικού μοντέλου (parallel machines)

4.4.Αντικειμενική Συνάρτηση

$$M \quad (\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{i=1}^{P}(X_{t,m,i}+Y_{t,m,i})\cdot C_{\ i}+\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{i=1}^{P}Y_{t,m,i}\cdot C_{\ i}+\sum_{t=1}^{T}\sum_{m=1}^{M}Z_{t,m}\cdot C_{\ m} \quad (1)$$

Η αντικειμενική συνάρτηση αποτελείται από τρείς συνιστώσες, οι οποίες αποτυπώνουν 1) το κόστος των εργατικών, 2) το κόστος εναλλαγής προϊόντων και 3) το κόστος πλυσιμάτων για όλες τις χρονικές περιόδους στον ορίζοντα προγραμματισμού και αναλύονται ως εξής:

1) **<u>Το κόστος εργατικών</u>**: Αποτυπώνει τις εργατοώρες που απαιτούνται για την εκπόνηση του προγράμματος παραγωγής και είναι το γινόμενο του σταθερού κόστους απασχόλησης μίας ομάδας 4 εργαζομένων για μια χρονική περίοδο με το πλήθος των χρονικών περιόδων για διεργασίες, που απαιτούν ανθρώπινο χειρισμό (Λειτουργία συσκευαστικών μηχανών και προετοιμασία συσκευαστικών μηχανών για αλλαγή προϊόντος). Το κόστος απασχόλησης σε αυτή τη θεώρηση είναι σταθερό σε όλη διάρκεια του ορίζοντα προγραμματισμού καθώς το πλήθος εξειδικευμένων χειριστών είναι συγκεκριμένο και το κόστος υπερωρίας ισούται με την προσαύξηση της βραδινής βάρδιας (ο χρόνος χειρισμού είναι ομοίως σταθερό μέγεθος καθώς προκύπτει από την ταχύτητα των μηχανών, η τιμή της οποίας είναι σταθερή.

Έτσι το γινόμενο κόστους υπερωριών επί του χρόνου ισούται με το αντίστοιχο γινόμενο βραδινών βαρδιών επί του χρόνου). Ταυτόχρονα στην ανάλυση κόστους παρατηρήσαμε ότι το μεγαλύτερο κόστος (άνω του 50%) είναι το κόστος πλυσιμάτων ενώ το εργατικό κόστος κυμαίνεται από 20% έως 30%, και συνεπώς θεωρώντας ένα μέσο εργατικό κόστος στο διευρυμένο μαθηματικό μοντέλο η λύση προσεγγίζει σε μεγάλο βαθμό το πρόβλημα.

2) **Το κόστος εναλλαγής προϊόντων**: Αποτυπώνει το κόστος εναλλαγής από προϊόν σε προϊόν στις συσκευαστικές μηχανές αναπαριστώντας τις απώλειες πρώτης ύλης για το άδειασμα των δεξαμενών εξισορρόπησης (buffer tanks) της κάθε μηχανής, ώστε αυτή να καταστεί έτοιμη για τη συσκευασία του επόμενου προϊόντος δίχως αποκλίσεις στην ποιότητα του (ποιοτικά χαρακτηριστικά, μετανάστευση χρώματος). Προκύπτει λοιπόν από το άθροισμα των γινομένων όλων των εναλλαγών που εκπονούνται με το κόστος της εκάστοτε εναλλαγής.
3) **Το κόστος πλυσιμάτων**: Αποτυπώνει το κόστος του πλυσίματος του εμπλεκόμενου εξοπλισμού καθώς και την ανάλωση των χημικών. Πρόκειται για το μεγαλύτερο τη τάξη κόστος καθώς περιλαμβάνει τις απώλειες πρώτης ύλης σε όλο το φάσμα της γραμμής παραγωγής (δεξαμενές, σωληνώσεις, μηχανές). Προκύπτει λοιπόν από το άθροισμα των γινόμενων του πλήθος των χρονικών περιόδων που λαμβάνει χώρα κάποιο πλύσιμο με το κόστος του εκάστοτε πλυσίματος.

4.5 Περιορισμοί

$$\sum_{i-1}^{P}(X_{t,m,i} + Y_{t,m,i}) + K_{t,m} + Z_{t,m} = 1 \quad \forall t, \forall m \quad (2)$$

Ο περιορισμός (2)εξασφαλίζει πως ανά πάσα στιγμή κάθε μηχανή θα είναι αυστηρά σε μία και μόνο κατάσταση (παραγωγή, προετοιμασία, αναμονή, πλύσιμο).

$$\sum_{m-1}^{M}\sum_{i-1}^{P}(X_{t,m,i} + Y_{t,m,i}) \leq 2 \quad \forall t \quad (3)$$

Ο περιορισμός (3) εξασφαλίζει πως ανά πάσα στιγμή θα λαμβάνουν χώρα το πολύ δύο διεργασίες, που απαιτούν παρουσία ομάδας εργαζομένων.

$$\sum_{t=1}^{T}\sum_{m=1}^{M} P_{m,i} \cdot X_{t,m,i} \geq D_i \; \forall i \quad (4)$$

Ο περιορισμός (4) εξασφαλίζει πως η ζήτηση κάθε προϊόντος θα καλυφθεί εντός του χρονικού ορίζοντα παραγωγής.

$$\sum_{t=1}^{T}\sum_{m=1}^{M} P_{m,i} \cdot X_{t,m,i} \leq D_i + P \quad \forall i \quad (5)$$

Ο περιορισμός (5) εξασφαλίζει πως η παραγωγή του κάθε προϊόντος δε θα ξεπεράσει τη ζήτηση κατά μεγάλο ποσοστό εντός του χρονικού ορίζοντα παραγωγής (καθώς η πλεονάζουσα ποσότητα δε μπορεί να διατεθεί στην αγορά).

$$Y_{t,m,i} \geq X_{t+1,m,i} - X_{t,m,i} \quad t = 1 \ldots . T-1, \forall m, \forall i \quad (6)$$

$$X_{t+1,m,i} - X_{t,m,i} + 1{,}1\left(1 - Y_{t+1,m,i}\right) \geq 0{,}1 \quad t = 1 \ldots . T-1, \forall m, \forall i \quad (7)$$

Ο συνδυασμός των περιορισμών (6) και (7) εξασφαλίζει πως κάθε μηχανή θα τεθεί σε κατάσταση προετοιμασίας πριν την έναρξη της παραγωγής οποιουδήποτε προϊόντος και θα ολοκληρώσει την προετοιμασίας της με την έναρξη της παραγωγής.

$$Z_{t+1,m} \geq \sum_{i=1}^{P}(X_{t,m,i} + Y_{t,m,i}) - \sum_{i=1}^{P}(X_{t+1,m,i} + Y_{t+1,m,i}) \quad t = 1 \ldots . T-1, \forall m, \forall i \quad (8)$$

Ο περιορισμός (8) εξασφαλίζει πως κάθε μηχανή θα τεθεί σε κατάσταση πλυσίματος μετά το πέρας κάποιας παραγωγικής δραστηριότητας.

$c_{t+1\,m}$ $Z_{t+1\,m}$ $Z_{t\,m}$ t T m

$Z_{t+1\,m}$ $Z_{t\,m}$ $($ $c_{t+1\,m})$ t T m

$$Z_{t,m} + Z_{t+1,m} \geq 2 \cdot c_{t,m} \quad t = 1 \ldots . T-1, \forall m \quad (11)$$

$\sum_{\iota=t}^{t+2} \sum_{i}^{P} X_{\tau}$ Y_{τ} t T m

$\sum_{t=1}^{T} \sum_{m=1}^{M} Y_{t\,m\,i}$ μ D P i

Περιορισμοί αλληλουχίας (levelling):

Free lact(k=0)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(X_{t-2,m,i}) + X_{t,m,k} \leq 1 \quad t = 3 \dots . T, m = 0,1 \qquad (14)$$

Ο περιορισμός (14) εξασφαλίζει πως το προϊόν Free lact δε θα έπεται της παραγωγής κάποιου άλλου προϊόντος στην ίδια μηχανή.

Split cup(k=6)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(X_{t-2,m,i}) + X_{t,m,k} \leq 1 \quad t = 3 \dots . T, m = 0 \qquad (15)$$

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,m,i}) + X_{t,m,k} \leq 1 \quad t = 1 \dots . T-1, m = 0 \qquad (16)$$

Ο συνδυασμός των περιορισμών (15) και (16) εξασφαλίζει πως το προϊόν Split cup δε θα έπεται ούτε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος στην ίδια μηχανή.

Set(k=13,14,15,16)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(X_{t-2,m,i}) + \sum_{k=1}^{k=1} X_{t,m,k} \leq 1 \quad t = 3 \dots . T, m = 1 \qquad (17)$$

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,m,i}) + \sum_{k=1}^{k=1} X_{t,m,k} \leq 1 \quad t = 1 \dots . T-1, m = 1 \qquad (18)$$

Ο συνδυασμός των περιορισμών (17) και (18) εξασφαλίζει πως τα προϊόντα τύπου set δε θα έπονται ούτε θα προηγούνται της παραγωγής προϊόντος άλλου τύπου στην ίδια μηχανή.

Κεφίρ (k=12)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,m,i}) + X_{t,m,k} \leq 1 \quad t = 1 \dots . T-1, m = 0,1 \qquad (19)$$

Ο περιορισμός (19) εξασφαλίζει πως το προϊόν Κεφίρ δε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος στην ίδια μηχανή.

Παιδικό Μπισκότο (k=19)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,m,i}) + X_{t,m,k} \leq 1 \quad t = 1 \ldots . T-1, m = 3 \qquad (20)$$

Ο περιορισμός (20) εξασφαλίζει πως το προϊόν Παιδικό Μπισκότο δε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος στην ίδια μηχανή.

Στραγγιστό mix(k=10)

$$\sum_{\substack{i=1 \\ i\neq k}}^{P}(Y_{t+1,m,i}) + X_{t,m,k} \leq 1 \quad t = 1 \ldots . T-1, m = 3 \qquad (21)$$

Ο περιορισμός (21) εξασφαλίζει πως το προϊόν Στραγγιστό Mix δε θα προηγείται της παραγωγής κάποιου άλλου προϊόντος στην ίδια μηχανή.

<u>Αρχικές και τελικές καταστάσεις:</u>

$$X_{t,m,i} = 0 \quad t = 0, \forall m, \forall i \qquad (22)$$

Ο περιορισμός (22) εξασφαλίζει πως κανένα προϊόν δε θα παράγεται σε καμία μηχανή κατά την αρχική περίοδο του ορίζοντα προγραμματισμού.

$$X_{T-1,m,i} + X_{T,m,i} + Y_{T-1,m,i} + Y_{T,m,i} = 0 \quad \forall m, \forall i \qquad (23)$$

Ο περιορισμός (23) εξασφαλίζει πως καμία παραγωγική δραστηριότητα δε θα λαμβάνει χώρα σε καμία μηχανή κατά τις δύο τελικές χρονικές περιόδους του ορίζοντα προγραμματισμού (έτσι ώστε οποιαδήποτε μηχανή να μπορεί να πλυθεί ώστε να είναι έτοιμη να τεθεί σε κάποια παραγωγική δραστηριότητα στην αρχή του επόμενου ορίζοντα).

4.6 Εφαρμογές του διευρυμένου μαθηματικού μοντέλου (parallel machines)

4.6.1 Εφαρμογή μοντέλου σε 5 μηχανές για 20 προϊόντα

4.6.1.1 Υπολογιστικά αποτελέσματα

Το κωδικοποιημένο μοντέλο λύθηκε σε υπολογιστικό περιβάλλον CPLEX_Studio125 με χρήση γλώσσα προγραμματισμού C++ σε ηλεκτρονικό υπολογιστή με χαρακτηριστικάIntel®Core™i5-3330 CPU @ 3.00GHz, RAM 8,00 GB παραθέτοντας τα παρακάτω αποτελέσματα:

Solution Value	36568
Time(sec)	3546(59,1 min)
Gap	53,5%

Πίνακας 14: Υπολογιστικό αποτέλεσμα μοντέλου (parallel machines)

Για να κατανοήσουμε σε βάθος το αποτέλεσμα της λύσης και να αντλήσουμε τα απαραίτητα συμπεράσματα οφείλουμε να αναλύσουμε τους κοστολογικούς όρους της αντικειμενικής συνάρτησης, την αλληλουχία καταστάσεων στη μηχανή καθώς και τη σειρά εκτέλεσης της παραγωγής των προϊόντων.

4.6.1.2 Αξιολόγηση αποτελεσμάτων

Όπως προαναφέραμε η αντικειμενική συνάρτηση αποτελείται από τρεις ομαδοποιημένους κοστολογικούς όρους οι οποίοι αποτυπώνουν 1) Το κόστος εργατικών, 2) το κόστος εναλλαγής προϊόντων και 3) το κόστος πλυσιμάτων.

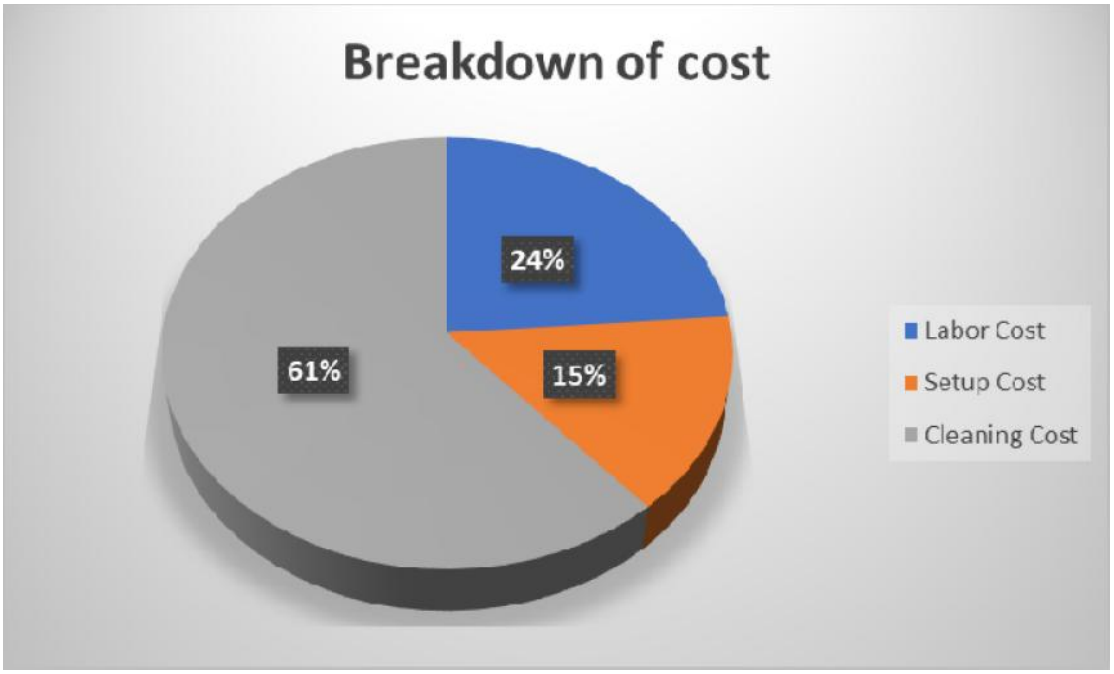
Στη λύση που προέκυψε αυτοί διαμοιράζονται ως εξής:

Costing terms	Cost (€)
Labor Cost	8700
Setup Cost	5428
Cleaning Cost	22440
Total Packaging Cost	36568

Πίνακας 15: Ανάλυση κοστολογικών όρων (parallel machines)

Total Production (kg)	298733,6
Labor Cost (€/ kg)	0,03
Total Pac Cost (€/ kg)	0,12

Πίνακας 16: Αναγωγή κοστολογικών όρων επί της παραγωγής (parallel machines.



Πίνακας 17: Ποσοστιαία αποτύπωση κοστολογικών όρων (parallel machines)

Επίσης θα ήταν χρήσιμο να υπολογίσουμε το ποσοστό απασχόλησης της κάθε μηχανής, καθώς αποτελεί δείκτη που αποτυπώνει τόσο τη αξιοποίηση του εξοπλισμού όσο και το περιθώριο κάλυψης αυξημένης ζήτησης. Το παρακάτω γράφημα παρουσιάζει τις περιόδους που βρίσκεται η κάθε μηχανή στην αντίστοιχη κατάσταση (X: Παραγωγή ,Y: Προετοιμασία ,Z: Πλύσιμο ,K: Αδράνεια):

	X	Y	Z	Machines	Utilization rate
150-450g M.1	75	5	10	150-450g M.1	54%
150-450g M.2	74	9	10	150-450g M.2	55%
1000g Machine	37	3	4	1000g Machine	26%
135g Machine	36	3	4	135g Machine	26%
5000g Machine	7	1	2	5000g Machine	6%
				Total Utilization	30%

	MONDAY																								TUESDAY																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
150-450g M.1	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
150-450g M.2	K	K	Y	X	X	X	Y	X	X	X	X	X	Y	X	X	X	X	X	Y	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
1000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Y	X	Z	Z	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

	WEDNESDAY																								THURSDAY																							
	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
150-450g M.1	X	X	X	X	X	Z	Z	K	Y	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z
150-450g M.2	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	Y	X	X	X	X	X	X	Y	X	X
1000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

	FRIDAY																							
	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
150-450g M.1	Z	Y	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K
150-450g M.2	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K
1000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	Z

	SATURDAY																							
	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
150-450g M.1	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
150-450g M.2	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X
1000g Machine	X	X	X	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K
135g Machine	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z
5000g Machine	Z	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

	SUNDAY																							
	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
150-450g M.1	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
150-450g M.2	X	X	X	X	X	Z	Z	K	K	K	K	Y	X	X	X	X	X	X	X	Z	Z	K	K	K
1000g Machine	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

4.6.2 Εφαρμογή μοντέλου σε 5 μηχανές για 20 προϊόντα (σενάριο αύξησης ταχύτητας μηχανών κατά 20%)

4.6.2.1 Υπολογιστικά αποτελέσματα

Το κωδικοποιημένο μοντέλο λύθηκε σε υπολογιστικό περιβάλλον CPLEX_Studio125 με χρήση γλώσσα προγραμματισμού C++ σε ηλεκτρονικό υπολογιστή με χαρακτηριστικά υπολογιστή Intel® Core™ i5-3330 CPU @ 3.00GHz, RAM 8,00 GB παραθέτοντας τα παρακάτω αποτελέσματα:

Solution Value	33717
Time(sec)	990(16,5 min)
Gap	55,8%

Πίνακας 19: Υπολογιστικό αποτέλεσμα μοντέλου (Increased Velocity)

Για να κατανοήσουμε σε βάθος το αποτέλεσμα της λύσης και να αντλήσουμε τα απαραίτητα συμπεράσματα οφείλουμε να αναλύσουμε τους κοστολογικούς όρους της αντικειμενικής συνάρτησης, την αλληλουχία καταστάσεων στη μηχανή καθώς και τη σειρά εκτέλεσης της παραγωγής των προϊόντων.

4.6.2.2 Αξιολόγηση αποτελεσμάτων

Όπως προαναφέραμε η αντικειμενική συνάρτηση αποτελείται από τρεις ομαδοποιημένους κοστολογικούς όρους οι οποίοι αποτυπώνουν 1) Το κόστος εργατικών, 2) το κόστος εναλλαγής προϊόντων και 3) το κόστος πλυσιμάτων.

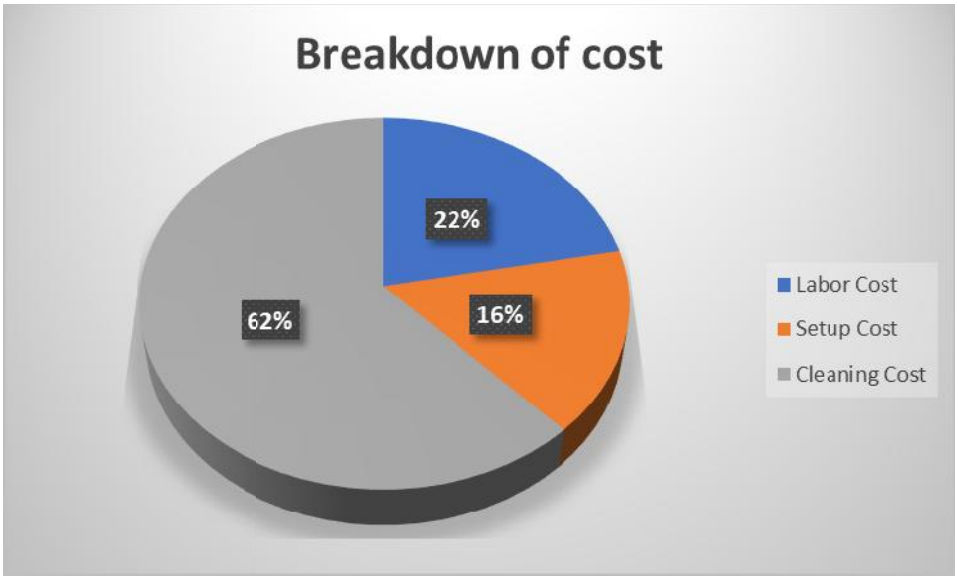
Στη λύση που προέκυψε αυτοί διαμοιράζονται ως εξής:

Costing terms	Cost (€)
Labor Cost	7377,6
Setup Cost	5428
Cleaning Cost	20912
Total Packaging Cost	33717,6

Πίνακας 20: Ανάλυση κοστολογικών όρων (Increased velocity)

Total Production (kg)	298733,6
Labor Cost (€/ kg)	0,02
Total Pac Cost (€/ kg)	0,11

Πίνακας 21: Αναγωγή κοστολογικών όρων επί της παραγωγής (Increased velocity)



Πίνακας 22: Ποσοστιαία αποτύπωση κοστολογικών όρων (Increased velocity)

Επίσης θα ήταν χρήσιμο να υπολογίσουμε το ποσοστό απασχόλησης της κάθε μηχανής, καθώς αποτελεί δείκτη που αποτυπώνει τόσο τη αξιοποίηση του εξοπλισμού όσο και το περιθώριο κάλυψης αυξημένης ζήτησης. Το παρακάτω γράφημα παρουσιάζει τις περιόδους που βρίσκεται η κάθε μηχανή στην αντίστοιχη κατάσταση (X: Παραγωγή ,Y: Προετοιμασία ,Z: Πλύσιμο ,K: Αδράνεια):

	X	Y	Z
150-450g M.1	59	5	8
150-450g M.2	65	9	10
1000g Machine	30	3	4
135g Machine	31	3	4
5000g Machine	6	1	2

Machines	Utilization rate
Gasti new	43%
Gasti old	50%
Alfa machine	22%
Waldner	23%
Miro 5	5%
Total Utilization	25%

	MONDAY																								TUESDAY																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
150-450g M.1	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
150-450g M.2	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	Y	X	X	X	Z	Z	K	K	K	K	K	K
1000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

	WEDNESDAY																								THURSDAY																							
	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
150-450g M.1	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	Y	X	X	X	X	X	X
150-450g M.2	K	K	K	K	K	K	K	Y	X	X	X	X	Y	X	X	X	X	Y	X	X	X	X	Z	Z	Y	X	X	X	X	X	X	X	X	X	Y	X	X	X	X	X	X	X	X	Z	Z	K	K	K
1000g Machine	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

	FRIDAY																									SATURDAY																								
	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
150-450g M.1	X	X	X	X	X	X	X	X	Z	Z	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	Y	X	X	X
150-450g M.2	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	Y	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
1000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
135g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	X	Y	X		
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

	SUNDAY																								
	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
150-450g M.1	X	X	X	X	X	X	X	X	X	X	X	Z	Z	K	K	K	K	K	K	K	K	K	K	K
150-450g M.2	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
1000g Machine	K	K	K	K	K	Y	X	X	X	X	X	X	X	X	X	X	X	Y	X	X	X	Z	Z	K
135g Machine	Z	Z	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
5000g Machine	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	Y	X	X	X	X	X	X	Z	Z

Normal Scenario		Upgrades Scenario	
Machines	**Utilization rate**	**Machines**	**Utilization rate**
150-450g M.1	54%	**150-450g M.1**	43%
150-450g M.2	55%	**150-450g M.2**	50%
1000g Machine	26%	**1000g Machine**	22%
135g Machine	26%	**135g Machine**	23%
5000g Machine	6%	**5000g Machine**	5%
Total Utilization	**30%**	**Total Utilization**	**25%**
		Total Capacity Incr.	**20%**
		Small Cups Capacity Incr.	**20%**

Total Packaging Cost (€)	36568	**Total Packaging Cost (€)**	33717,6
		Weekly Profit Increase (€)	**2850,4**
		Total Pac. Cost decrease	**-7,79%**

5. ΣΥΜΠΕΡΑΜΑΤΑ

Στην παρούσα εργασία παρουσιάστηκαν δύο μοντέλα μεικτού γραμμικού-ακέραιου προγραμματισμού με διακριτή θεώρηση χρόνου και εφαρμόστηκαν στο παράδειγμα μιας γαλακτοβιομηχανίας με τους περιορισμούς που διέπουν τις διαδικασίες της. Η διακριτή αυτή θεώρηση του χρόνου ενισχύει την πρακτικότητα του μοντέλου καθώς ο κώδικας βελτιστοποίησης μπορεί να εφαρμοστεί ανά πάσα στιγμή. Αύτη η ευελιξία του μοντέλου είναι που καταδεικνύει και τη χρησιμότητα του καθώς το ζήτημα του χρονοπρογραμματισμού της παραγωγής εκτίθεται σε μία σειρά παραγόντων αβεβαιότητας και έκτακτων συμβάντων-αναθεωρήσεων, όπως για παράδειγμα οι μηχανικές βλάβες (breakdowns), η ακύρωση παραγγελιών (order cancellation), απουσία εργαζόμενων και πληθώρα περιστατικών, που μπορεί να μεταβάλουν τα δεδομένα του προβλήματος.

Το απλό μοντέλο της μιας μηχανής προσφέρει ακριβέστερα αποτελέσματα στο επίπεδο της μιας μηχανής καθώς λαμβάνουμε υπόψη και τον παράγοντα κόστους εργατοωρών σε βραδινή βάρδια και σε αργίες. Αντιθέτως στην παρούσα φάση δεν μπορεί να προστεθεί στο υπολογιστικό μοντέλο του διευρυμένου προβλήματος, ο παράγοντας του διαφορετικού κόστους εργατοώρας καθώς γιγαντώνεται ακόμη περισσότερο ο υπολογιστικός φόρτος με αποτέλεσμα να μας δίνει μια λύση που απέχει αρκετά από την βέλτιστη (μεγάλο Gap). Το μοντέλο της μίας μηχανής αδυνατεί όμως να λειτουργήσει συνδυαστικά ενώ οι ανάγκες της εταιρείας λόγω της ανάπτυξής της επιτάσσουν να δουλεύει συνδυαστικά ο εξοπλισμός της.

Το διευρυμένο μοντέλο για την αποτύπωση των πραγματικών συνθηκών παραγωγής μπορεί με μικρές διαφοροποιήσεις να εξετάσει τόσο την εφικτότητα σεναρίων που επηρεάζουν ριζικά το σχεδιασμό παραγωγής, όπως η κατακόρυφη αύξηση της ζήτησης, όσο και τη βελτιστότητα-επίδραση στρατηγικών αποφάσεων όπως επενδύσεις επέκτασης των γραμμών παραγωγής.

Απώτερος στόχος της εργασίας είναι το μοντέλο να αποτελέσει βασικό εργαλείο προγραμματισμού της παραγωγής με τη βοήθεια υπολογιστικών συστημάτων, που θα μπορέσει να εφαρμοστεί συνδυαστικά με άλλα λειτουργικά συστήματα ανάλυσης αποθεμάτων (Warehouse Management), διορατικά συστήματα (Mrp, Mps), ώστε να παρέχει πολυκριτηριακή ανάλυση και βελτιστοποίηση στη λήψη αποφάσεων.

Αντικείμενο λοιπόν, μελλοντικής μελέτης μπορεί να αποτελέσει η βελτίωση του υπολογιστικού μοντέλου και ίσως και του μαθηματικού μοντέλου με προσθήκη περιορισμού που θα επιταχύνουν την λύση στα πλαίσια τις μείωσης του υπολογιστικού χρόνου και η διασύνδεση του με άλλες μεθόδους λήψης αποφάσεων. Τέλος, η εφαρμογή του μπορεί να επεκταθεί στην παραγωγή πληθώρας προϊόντων, που χαρακτηρίζονται από όμοιους περιορισμούς διάρκειας ζωής.

Pmi	150-450g M.1	Di	Csi	Pmin	Ccm	Cl
FREE LACT 150g	8000	76475	296	84475	764	34,8
ΣΤΡΑΓΓΙΣΤΟ 10% 200g	10000	161195	276	171195		43,5
ΣΤΡΑΓΓΙΣΤΟ 2% 200g	10000	193115	270	203115		60,9
ΣΤΡΑΓΓΙΣΤΟ 2% split cup	3600	25000	270	28600		
ΣΤΡΑΓΓΙΣΤΟ 0% 200g	10000	126025	264	136025		
ΣΤΡΑΓΓΙΣΤΟ 0% UNDER 150g	6000	37000	314	43000		
ΣΤΡΑΓΓΙΣΤΟ 0% MIX 150g	6000	236000	364	242000		
ΠΝΟΗ 170g	8000	41175	196	49175		
KEFIR 150g	6000	64392	324	70392		

Pmi	150-450g M.1	150-450g M.2	1000g Machine	135g Machine	5000g Machine	150-450g New	Di	Csi	Pmin	Ccm	Cl
FREE LACT 150g	8000	8000	0	0	0	8000	76475	296	84475	764	34,8
ΣΤΡΑΓΓΙΣΤΟ 10% 5kg	0	0	0	0	1575	0	10500	132	12075	764	
ΣΤΡΑΓΓΙΣΤΟ 10% 1kg	0	0	2700	0	0	0	33480	220	36180	648	
ΣΤΡΑΓΓΙΣΤΟ 10% 200g	10000	10000	0	0	0	10000	161195	276	171195	944	
ΣΤΡΑΓΓΙΣΤΟ 2% 1kg	0	0	2700	0	0	0	51840	216	54540	396	
ΣΤΡΑΓΓΙΣΤΟ 2% 200g	10000	10000	0	0	0	10000	193115	270	203115	764	
ΣΤΡΑΓΓΙΣΤΟ 2% split cup	3600	0	0	0	0	0	25000	270	28600		
ΣΤΡΑΓΓΙΣΤΟ 0% 1kg	0	0	2700	0	0	0	9510	212	12210		
ΣΤΡΑΓΓΙΣΤΟ 0% 200g	10000	10000	0	0	0	10000	126025	264	136025		
ΣΤΡΑΓΓΙΣΤΟ 0% UNDER 150g	6000	6000	0	0	0	6000	37000	314	43000		
ΣΤΡΑΓΓΙΣΤΟ 0% MIX 150g	6000	0	0	0	0	6000	236000	364	242000		
ΠΝΟΗ 170g	8000	8000	0	0	0	8000	41175	196	49175		
KEFIR 150g	6000	6000	0	0	0	6000	64392	324	70392		
ΚΑΤΣΙΚΙΣΙΟ 200g	0	6000	0	0	0	0	28610	188	34610		
ΠΡΟΒΕΙΟ 200g	0	6000	0	0	0	0	15000	246	21000		
ΑΓΕΛΑΔΟΣ 2% 200g	0	6000	0	0	0	0	26170	116	32170		
ΒΙΟΛΟΓΙΚΟ 200g	0	6000	0	0	0	0	25160	206	31160		
ΠΑΙΔΙΚΟ ΜΠΑΝΑΝΑ+ΒΑΝ	0	0	0	2500	0	0	39760	292	42260		
ΠΑΙΔΙΚΟ ΦΡΑΟΥΛΑ+ΚΕΡ	0	0	0	2500	0	0	45300	308	47800		
ΠΑΙΔΙΚΟ ΜΠΙΣΚΟΤΟ	0	0	0	2500	0	0	2496	354	4996		

6.2 Κωδικοποίηση μοντέλου

Single machine C++ coding:

```
/*Optimal scheduling in a yogurt production line(Single Machine)*/
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

#include <vector>
#include <string>
#include <sstream>
#include <fstream>
#include <stdlib.h>
using std::vector;


int main(){
    /*------------------*
     *                  *
     *   Definitions    *
     *                  *
     *------------------*/
    int         t,m,i;                  // t:period,m:machine,i:product
    const int   tmax=168;               // period max
    const int   imax=9;                 // product max
    const double    Cl1=34.8;           // labor cost of a single shift in a single period
    const double    Cl2=43.5;           // evening labor cost of a single shift in a single period
    const double    Cl3=60.9;           // holiday labor cost of a single shift in a single period
    const double    Cc=573;             // cleaning cost of machine in a single period

    /* TABLES */
    int         D[imax];            // demand of product
    int         Pi[imax];           // production of product i in machine in a single period
    int         Pmin[imax];         // Upper bound for production(D[i]+P[i])
    int         Cs[imax];           // setup cost for product i

    /*------------------*
     *                  *
     *  Initialization  *
     *                  *
     *------------------*/

    for (int x=0; x<imax; x++){
        D[x]=0;
        Pi[x]=0;
        Pmin[x]=0;
        Cs[x]=0;
    }

    /*------------------*
     *                  *
     *      Inputs      *
     *                  *
     *------------------*/

    ifstream ifs_1;
    ifs_1.open("Demand_gasti.txt");

    for (int j=0;j<imax;j++){
        ifs_1 >> D[j];
    }
```

```
58.
59.     ifstream ifs_2;
60.     ifs_2.open("Csi_gasti.txt");
61.
62.     for (int j=0;j<imax;j++){
63.         ifs_2 >> Cs[j];
64.     }
65.
66.     ifstream ifs_3;
67.     ifs_3.open("Pmin_gasti.txt");
68.
69.     for (int j=0;j<imax;j++){
70.         ifs_3 >> Pmin[j];
71.     }
72.
73.     ifstream ifs_4;
74.     ifs_4.open("Pmi_gasti.txt");
75.
76.     for (int j=0;j<imax;j++){
77.         ifs_4 >> Pi[j];
78.     }
79.     /*------------------*
80.      *                  *
81.      *      Model       *
82.      *                  *
83.      *------------------*/
84.
85.     IloEnv env;
86.     try{
87.         IloModel model  (env);
88.
89.         typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
90.
91.         typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
92.
93.         IloCplex cplex(env);
94.         // Gap-Time Limit
95.         cplex.setParam(IloCplex::EpGap, 0.2);
96.         //cplex.setParam(IloCplex::TiLim, 600);
97.
98.         /*----------------------*
99.          *                      *
100.                *  Decision Variables  *
101.                *                      *
102.                *----------------------*/
103.
104.               // X[ti]
105.               IloNumVarMatrix2x2 Xti(env,0);
106.               for (int k=0;k<tmax;k++){
107.                   IloNumVarArray Xi(env,0);
108.                   for (int d=0;d<imax;d++){
109.                       char Path[70];
110.                       sprintf(Path,"Xti(t%d,i%d)",k,d);
111.                       IloNumVar X(env,0,1,ILOBOOL,Path);
112.                       Xi.add(X);
113.                   }
114.                   Xti.add(Xi);
115.               }
116.
117.               // Y[ti]
118.               IloNumVarMatrix2x2 Yti(env,0);
119.               for (int k=0;k<tmax;k++){
120.                   IloNumVarArray Yi(env,0);
121.                   for (int d=0;d<imax;d++){
122.                       char Path[70];
```

```
123.                     sprintf(Path,"Yti(t%d,i%d)",k,d);
124.                     IloNumVar Y(env,0,1,ILOBOOL,Path);
125.                     Yi.add(Y);
126.                 }
127.                 Yti.add(Yi);
128.             }
129.
130.             // Z[t]
131.             IloNumVarArray Zt(env,0);
132.             for (int d=0;d<tmax;d++){
133.                 char Path[70];
134.                 sprintf(Path,"Zt(t%d",d);
135.                 IloNumVar Z(env,0,1,ILOBOOL,Path);
136.                 Zt.add(Z);
137.             }
138.
139.             // K[tm]
140.             IloNumVarArray Kt(env,0);
141.             for (int d=0;d<tmax;d++){
142.                 char Path[70];
143.                 sprintf(Path,"Kt(t%d",d);
144.                 IloNumVar K(env,0,1,ILOBOOL,Path);
145.                 Kt.add(K);
146.             }
147.
148.             // c[t]
149.             IloNumVarArray ct(env,0);
150.             for (int d=0;d<tmax;d++){
151.                 char Path[70];
152.                 sprintf(Path,"ct(t%d",d);
153.                 IloNumVar c(env,0,1,ILOBOOL,Path);
154.                 ct.add(c);
155.             }
156.
157.
158.             /*------------------*
159.              *                  *
160.              *   Constraints    *
161.              *                  *
162.              *------------------*/
163.
164.             // Constraint 2
165.             IloRangeArray Constraint2_Ar(env,0);
166.             for (int x2 = 0; x2 < tmax; x2++){
167.                 IloExpr expr(env,0);
168.                 for (int x3 = 0; x3 < imax; x3++){
169.                     expr += Xti[x2][x3]+Yti[x2][x3];
170.                 }
171.                 expr += Kt[x2] + Zt[x2];
172.
173.                 int LB=1, UB=1;
174.                 IloRange Constraint2_(env,LB,expr,UB);
175.                 expr.end();
176.                 model.add(Constraint2_);
177.                 Constraint2_Ar.add(Constraint2_);
178.             }
179.
180.             // Constraint 3-4
181.             IloRangeArray Constraint3_Ar(env,0);
182.             for (int x1 = 0; x1 < imax; x1++){
183.                 IloExpr expr(env,0);
184.                 for (int x2 = 0; x2 < tmax; x2++){
185.                     expr += Pi[x1]*Xti[x2][x1];
186.                 }
187.
```

```
188.                    float LB=D[x1], UB=Pmin[x1];
189.                    IloRange Constraint3_(env,LB,expr,UB);
190.                    expr.end();
191.                    model.add(Constraint3_);
192.                    Constraint3_Ar.add(Constraint3_);
193.                }
194.
195.                // Constraint 5
196.                IloRangeMatrix2x2 Constraint5_2x2(env,0);
197.                for (int x2 = 0; x2 < tmax-1; x2++){
198.                    IloRangeArray Constraint5_Ar(env,0);
199.                    for (int x3 = 0; x3 < imax; x3++){
200.                        IloExpr expr(env,0);
201.                        expr = Yti[x2][x3]-Xti[x2+1][x3]+Xti[x2][x3];
202.
203.                        float LB=0, UB=IloInfinity;
204.                        IloRange Constraint5_(env,LB,expr,UB);
205.                        expr.end();
206.                        model.add(Constraint5_);
207.                        Constraint5_Ar.add(Constraint5_);
208.                    }
209.                    Constraint5_2x2.add(Constraint5_Ar);
210.                }
211.
212.                // Constraint 6
213.                IloRangeMatrix2x2 Constraint6_2x2(env,0);
214.                for (int x2 = 0; x2 < tmax-1; x2++){
215.                    IloRangeArray Constraint6_Ar(env,0);
216.                    for (int x3 = 0; x3 < imax; x3++){
217.                        IloExpr expr(env,0);
218.                        expr = Xti[x2+1][x3] - Xti[x2][x3] + 1.1 * (1 - Yti[x2][x3] );
219.
220.                        float LB=0.1, UB=IloInfinity;
221.                        IloRange Constraint6_(env,LB,expr,UB);
222.                        expr.end();
223.                        model.add(Constraint6_);
224.                        Constraint6_Ar.add(Constraint6_);
225.                    }
226.                    Constraint6_2x2.add(Constraint6_Ar);
227.                }
228.
229.                // Constraint 7
230.                IloRangeArray Constraint7_Ar(env,0);
231.                for (int x2 = 0; x2 < tmax-1; x2++){
232.                    IloExpr expr(env,0);
233.                    for (int x3 = 0; x3 < imax; x3++){
234.                        expr -= Xti[x2][x3] + Yti[x2][x3] - Xti[x2+1][x3] - Yti[x2+1][x3];
235.                    }
236.
237.                    expr += Zt[x2+1];
238.                    float LB=0, UB=IloInfinity;
239.                    IloRange Constraint7_(env,LB,expr,UB);
240.                    expr.end();
241.                    model.add(Constraint7_);
242.                    Constraint7_Ar.add(Constraint7_);
243.                }
244.
245.                // Constraint 8
246.                IloRangeArray Constraint8_Ar(env,0);
247.                for (int x2 = 0; x2 < tmax-1; x2++){
248.                    IloExpr expr(env,0);
249.                    expr = ct[x2+1]-Zt[x2+1]+Zt[x2];
250.
251.                    float LB=0, UB=1;
252.                    IloRange Constraint8_(env,LB,expr,UB);
```

```
253.                    expr.end();
254.                    model.add(Constraint8_);
255.                    Constraint8_Ar.add(Constraint8_);
256.                }
257.
258.                // Constraint 9
259.                IloRangeArray Constraint9_Ar(env,0);
260.                for (int x2 = 0; x2 < tmax-1; x2++){
261.                    IloExpr expr(env,0);
262.                    expr = Zt[x2+1] - Zt[x2] + 1.1 * ( 1 - ct[x2+1] );
263.
264.                    float LB=0.1, UB=IloInfinity;
265.                    IloRange Constraint9_(env,LB,expr,UB);
266.                    expr.end();
267.                    model.add(Constraint9_);
268.                    Constraint9_Ar.add(Constraint9_);
269.                }
270.
271.                // Constraint 10
272.                IloRangeArray Constraint10_Ar(env,0);
273.                for (int x2 = 0; x2 < tmax-1; x2++){
274.                    IloExpr expr(env,0);
275.
276.                    expr = Zt[x2]+Zt[x2+1]- 2*ct[x2];
277.                    float LB=0, UB=1;
278.                    IloRange Constraint10_(env,LB,expr,UB);
279.                    expr.end();
280.                    model.add(Constraint10_);
281.                    Constraint10_Ar.add(Constraint10_);
282.                }
283.
284.                // Constraint 11
285.                IloRangeArray Constraint11_Ar(env,0);
286.                for (int x1 = 0; x1 < tmax-23; x1++){
287.                    IloExpr expr(env,0);
288.                    for (int x3 = x1; x3 < x1+23; x3++){
289.                        for (int x4 = 0; x4 < imax; x4++){
290.                            expr += Xti[x3][x4] + Yti[x3][x4];
291.                        }
292.                    }
293.
294.                    int LB=0, UB=22;
295.                    IloRange Constraint11_(env,LB,expr,UB);
296.                    expr.end();
297.                    model.add(Constraint11_);
298.                    Constraint11_Ar.add(Constraint11_);
299.                }
300.
301.                //Constraint12
302.                IloRangeArray Constraint12_Ar(env,0);
303.                for (int x1 = 0; x1 < imax; x1++){
304.                    if ((D[x1]-22*Pi[x1])<=0){
305.                        IloExpr expr(env,0);
306.                        for (int x2 = 0; x2 < tmax; x2++){
307.                            expr += Yti[x2][x1];
308.                        }
309.                        float LB=0, UB=1;
310.                        IloRange Constraint12_(env,LB,expr,UB);
311.                        expr.end();
312.                        model.add(Constraint12_);
313.                        Constraint12_Ar.add(Constraint12_);
314.                    }
315.
316.                    else{
317.                        IloExpr expr(env,0);
```

```
318.                     for (int x2 = 0; x2 < tmax; x2++){
319.                         expr += Yti[x2][x1];
320.                     }
321.                     float LB=0, UB=3;
322.                     IloRange Constraint12_(env,LB,expr,UB);
323.                     expr.end();
324.                     model.add(Constraint12_);
325.                     Constraint12_Ar.add(Constraint12_);
326.                 }
327.             }
328.
329.             //Constraint 13(free lact)
330.             IloRangeArray Constraint13_Ar(env,0);
331.             for (int x1 = 2; x1 < tmax; x1++){
332.                 IloExpr expr(env,0);
333.                 for (int x3 = 0; x3 < imax; x3++){
334.                     if (x3==0) continue;
335.                     expr += Xti[x1-2][x3];
336.                 }
337.                 expr += Xti[x1][0];
338.                 int LB=0 , UB=1;
339.                 IloRange Constraint13_(env,LB,expr,UB);
340.                 expr.end();
341.                 model.add(Constraint13_);
342.                 Constraint13_Ar.add(Constraint13_);
343.             }
344.
345.             //Constraint 14(split)
346.             IloRangeArray Constraint14_Ar(env,0);
347.             for (int x1 = 2; x1 < tmax; x1++){
348.                 IloExpr expr(env,0);
349.                 for (int x3 = 0; x3 < imax; x3++){
350.                     if (x3==3) continue;
351.                     expr += Xti[x1-2][x3];
352.                 }
353.                 expr += Xti[x1][3];
354.                 int LB=0 , UB=1;
355.                 IloRange Constraint14_(env,LB,expr,UB);
356.                 expr.end();
357.                 model.add(Constraint14_);
358.                 Constraint14_Ar.add(Constraint14_);
359.             }
360.
361.             //Constraint 15(split)
362.             IloRangeArray Constraint15_Ar(env,0);
363.             for (int x1 = 0; x1 < tmax-1; x1++){
364.                 IloExpr expr(env,0);
365.                 for (int x3 = 0; x3 < imax; x3++){
366.                     if (x3==3) continue;
367.                     expr += Yti[x1+1][x3];
368.                 }
369.                 expr += Xti[x1][3];
370.                 int LB=0 , UB=1;
371.                 IloRange Constraint15_(env,LB,expr,UB);
372.                 expr.end();
373.                 model.add(Constraint15_);
374.                 Constraint15_Ar.add(Constraint15_);
375.             }
376.
377.             //Constraint 16(kefir)
378.             IloRangeArray Constraint16_Ar(env,0);
379.             for (int x1 = 0; x1 < tmax - 1; x1++){
380.                 IloExpr expr(env,0);
381.                 for (int x3 = 0; x3 < imax; x3++){
382.                     if (x3==8) continue;
```

```
383.                     expr += Yti[x1+1][x3];
384.                 }
385.                 expr += Xti[x1][8];
386.                 int LB=0 , UB=1;
387.                 IloRange Constraint16_(env,LB,expr,UB);
388.                 expr.end();
389.                 model.add(Constraint16_);
390.                 Constraint16_Ar.add(Constraint16_);
391.             }
392.
393.             //Constraint 17(mix)
394.             IloRangeArray Constraint17_Ar(env,0);
395.             for (int x1 = 0; x1 < tmax - 1; x1++){
396.                 IloExpr expr(env,0);
397.                 for (int x3 = 0; x3 < imax; x3++){
398.                     if (x3==6) continue;
399.                     expr += Yti[x1+1][x3];
400.                 }
401.                 expr += Xti[x1][6];
402.                 int LB=0 , UB=1;
403.                 IloRange Constraint17_(env,LB,expr,UB);
404.                 expr.end();
405.                 model.add(Constraint17_);
406.                 Constraint17_Ar.add(Constraint17_);
407.             }
408.
409.             // Constraint 18
410.             IloRangeArray Constraint18_Ar(env,0);
411.             for (int x2 = 0; x2 < imax; x2++){
412.                 IloExpr expr(env,0);
413.                 expr = Xti[0][x2];
414.
415.                 int LB=0, UB=0;
416.                 IloRange Constraint18_(env,LB,expr,UB);
417.                 expr.end();
418.                 model.add(Constraint18_);
419.                 Constraint18_Ar.add(Constraint18_);
420.             }
421.
422.             // Constraint 19
423.             IloRangeArray Constraint19_Ar(env,0);
424.             for (int x2 = 0; x2 < imax; x2++){
425.                 IloExpr expr(env,0);
426.                 expr = Xti[166][x2]+Yti[166][x2]+Xti[167][x2]+Yti[167][x2];
427.
428.                 int LB=0, UB=0;
429.                 IloRange Constraint19_(env,LB,expr,UB);
430.                 expr.end();
431.                 model.add(Constraint19_);
432.                 Constraint19_Ar.add(Constraint19_);
433.             }
434.
435.             // Constraint20
436.             IloExpr expr(env,0);
437.             for (int x1 = 0; x1 < tmax; x1++){
438.                 expr += Zt[x1];
439.             }
440.             int LB=0 , UB=20;
441.             IloRange Constraint20_(env,LB,expr,UB);
442.             expr.end();
443.             model.add(Constraint20_);
444.
445.
446.             /*------------------*
447.              *                  *
```

```
448.                 *  Objective Func  *
449.                 *                  *
450.                 *------------------*/
451.
452.                IloExpr expr_obj(env);
453.                for (int x1=0; x1<16; x1++){
454.                    for (int x3=0; x3<imax; x3++){
455.                        expr_obj += Xti[x1][x3]+Yti[x1][x3];
456.                    }
457.                }
458.                for (int x1=24; x1<40; x1++){
459.                    for (int x3=0; x3<imax; x3++){
460.                        expr_obj += Xti[x1][x3]+Yti[x1][x3];
461.                    }
462.                }
463.                for (int x1=48; x1<64; x1++){
464.                    for (int x3=0; x3<imax; x3++){
465.                        expr_obj += Xti[x1][x3]+Yti[x1][x3];
466.                    }
467.                }
468.                for (int x1=72; x1<88; x1++){
469.                    for (int x3=0; x3<imax; x3++){
470.                        expr_obj += Xti[x1][x3]+Yti[x1][x3];
471.                    }
472.                }
473.                for (int x1=96; x1<112; x1++){
474.                    for (int x3=0; x3<imax; x3++){
475.                        expr_obj += Xti[x1][x3]+Yti[x1][x3];
476.                    }
477.                }
478.                for (int x1=120; x1<136; x1++){
479.                    for (int x3=0; x3<imax; x3++){
480.                        expr_obj += Xti[x1][x3]+Yti[x1][x3];
481.                    }
482.                }
483.                expr_obj *= Cl1;
484.                //---------------------------Cl1-----------------------//
485.                for (int x1=16; x1<24; x1++){
486.                    for (int x3=0; x3<imax; x3++){
487.                        expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl2;
488.                    }
489.                }
490.                for (int x1=40; x1<48; x1++){
491.                    for (int x3=0; x3<imax; x3++){
492.                        expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl2;
493.                    }
494.                }
495.                for (int x1=64; x1<72; x1++){
496.                    for (int x3=0; x3<imax; x3++){
497.                        expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl2;
498.                    }
499.                }
500.                for (int x1=88; x1<96; x1++){
501.                    for (int x3=0; x3<imax; x3++){
502.                        expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl2;
503.                    }
504.                }
505.                for (int x1=112; x1<120; x1++){
506.                    for (int x3=0; x3<imax; x3++){
507.                        expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl2;
508.                    }
509.                }
510.                for (int x1=136; x1<144; x1++){
511.                    for (int x3=0; x3<imax; x3++){
512.                        expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl2;
```

```
513.                 }
514.             }
515.             //---------------------------Cl2----------------------//
516.             for (int x1=144; x1<168; x1++){
517.                 for (int x3=0; x3<imax; x3++){
518.                     expr_obj += (Xti[x1][x3]+Yti[x1][x3])*Cl3;
519.                 }
520.             }
521.             //---------------------------Cl3----------------------//
522.             for (int x1=0; x1<tmax; x1++){
523.                 for (int x3=0; x3<imax; x3++){
524.                     expr_obj += Yti[x1][x3] * Cs[x3];
525.                 }
526.             }
527.             //-------------------------Csi-------------------------//
528.             for (int x1=0; x1<tmax; x1++){
529.                 expr_obj += Zt[x1] * Cc;
530.             }
531.             model.add(IloMinimize(env, expr_obj));
532.             expr_obj.end();
533.
534.             /*------------------*
535.              *                  *
536.              *     Solution     *
537.              *                  *
538.              *------------------*/
539.
540.             // Solve
541.             cplex.extract(model);
542.             cplex.exportModel("sol.lp");
543.             // Print Results
544.             if (!cplex.solve ()){
545.                 env.error()<<"Faild to optimize LP."<<endl;
546.                 throw(-1);
547.             }
548.
549.             env.out()<<"Solution status = " <<cplex.getStatus()<<endl;
550.             env.out()<<"Solution value = " <<cplex.getObjValue()<<endl;
551.
552.
553.             cplex.solve();
554.
555.             ofstream myfile;
556.             myfile.open("output.txt");
557.             for (int j=0;j<tmax;j++){
558.                 int g = cplex.getValue(Kt[j]);
559.                 if(g!=0) {
560.                     std::ostringstream oss;
561.                     oss <<"Kt"<<"("<<j<<")"<<"="<<g<<endl;
562.                     std::string var = oss.str();
563.                     myfile << var;
564.                 }
565.                 g = cplex.getValue(Zt[j]);
566.                 if(g!=0) {
567.                     std::ostringstream oss;
568.                     oss <<"Zt"<<"("<<j<<")"<<"="<<g<<endl;
569.                     std::string var = oss.str();
570.                     myfile << var;
571.                 }
572.                 for (int d=0;d<imax;d++){
573.                     g = cplex.getValue(Xti[j][d]);
574.                     if(g!=0) {
575.                         std::ostringstream oss;
576.                         oss <<"Xti"<<"("<<j<<","<<d<<")"<<"="<<g<<endl;
577.                         std::string var = oss.str();
```

```
                            myfile << var;
                        }
                        g = cplex.getValue(Yti[j][d]);
                        if(g!=0) {
                            std::ostringstream oss;
                            oss <<"Yti"<<"("<<j<<","<<d<<")"<<"="<<g<<endl;
                            std::string var = oss.str();
                            myfile << var;
                        }
                    }
                }
                myfile.close();


            }
            catch ( IloException& e){
                cerr <<"concert exception caught:"<<e<<endl;
            }
            catch (...){
                cerr <<"Unknown exception caught"<<endl;
            }
            // End env
            env.end();

            system("pause");
            return 0;
        }
```

5 parallel machines C++ coding:

```
/*Optimal scheduling in a yogurt production line*/
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

#include <vector>
#include <string>
#include <sstream>
#include <fstream>
#include <stdlib.h>
using std::vector;


int main(){
    /*------------------*
     *                  *
     *   Definitions    *
     *                  *
     *------------------*/

    int         t,m,i;              // t:period,m:machine,i:product
    const int   tmax=168;           // period max
    const int   mmax=5;             // machine max
    const int   imax=20;            // product max
    const double    Cl=34.8;        // labor cost of a single shift in a single period

    /* TABLES */
    int         Cc[mmax];           //cleaning cost of machine
    int         D[imax];            // demand of product
    int         P[mmax][imax];      // production of product i in machine m in a single period
    int         Pi[imax];
```

```
31.     int         Pmin[imax];         // Upper bound for production(D[i]+P[i])
32.     int         Cs[imax];           // setup cost for product i
33.
34.     /*------------------*
35.      *                  *
36.      *  Initialization  *
37.      *                  *
38.      *------------------*/
39.
40.     for (int x=0; x<mmax; x++){
41.         Cc[x]=0;
42.         for (int j=0;j<imax;j++){
43.             P[x][j]=0;
44.         }
45.     }
46.     for (int x=0; x<imax; x++){
47.         D[x]=0;
48.         Cs[x]=0;
49.         Pmin[x]=0;
50.         Pi[x]=0;
51.     }
52.
53.     /*------------------*
54.      *                  *
55.      *      Inputs      *
56.      *                  *
57.      *------------------*/
58.
59.     ifstream ifs_1;
60.     ifs_1.open("Demand_revised_20.txt");
61.
62.     for (int j=0;j<imax;j++){
63.         ifs_1 >> D[j];
64.     }
65.
66.     ifstream ifs_2;
67.     ifs_2.open("Pmi_revised_20.txt");
68.
69.     for (int i=0;i<mmax;i++){
70.         for (int j=0;j<imax;j++){
71.             ifs_2 >> P[i][j];
72.         }
73.     }
74.
75.     ifstream ifs_3;
76.     ifs_3.open("Csi_revised_20.txt");
77.
78.     for (int j=0;j<imax;j++){
79.         ifs_3 >> Cs[j];
80.     }
81.
82.     ifstream ifs_4;
83.     ifs_4.open("Ccm_revised_20.txt");
84.
85.     for (int j=0;j<mmax;j++){
86.         ifs_4 >> Cc[j];
87.
88.     }
89.     ifstream ifs_5;
90.     ifs_5.open("Pmin_revised_20.txt");
91.
92.     for (int j=0;j<imax;j++){
93.         ifs_5 >> Pmin[j];
94.     }
95.
```

```
96.     ifstream ifs_6;
97.     ifs_6.open("Pi_revised_20.txt");
98.
99.     for (int j=0;j<imax;j++){
100.                ifs_6 >> Pi[j];
101.            }
102.
103.            /*------------------*
104.             *                  *
105.             *      Model       *
106.             *                  *
107.             *------------------*/
108.
109.            IloEnv env;
110.            try{
111.                IloModel model  (env);
112.
113.                typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
114.                typedef IloArray<IloNumVarMatrix2x2> IloNumVarMatrix3x3;
115.
116.                typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
117.                typedef IloArray<IloRangeMatrix2x2> IloRangeMatrix3x3;
118.
119.                IloCplex cplex(env);
120.                // Gap-Time limit
121.                cplex.setParam(IloCplex::EpGap, 0.2);
122.                //cplex.setParam(IloCplex::TiLim, 600);
123.
124.                /*----------------------*
125.                 *                      *
126.                 *  Decision Variables  *
127.                 *                      *
128.                 *----------------------*/
129.
130.                // X[tmi]
131.                IloNumVarMatrix3x3 Xtmi(env,0);
132.                for (int j=0;j<tmax;j++){
133.                    IloNumVarMatrix2x2 Xmi(env,0);
134.                    for (int k=0;k<mmax;k++){
135.                        IloNumVarArray Xi(env,0);
136.                        for (int d=0;d<imax;d++){
137.                            char Path[70];
138.                            sprintf(Path,"Xtmi(t%d,m%d,i%d)",j,k,d);
139.                            IloNumVar X(env,0,1,ILOBOOL,Path);
140.                            Xi.add(X);
141.                        }
142.                        Xmi.add(Xi);
143.                    }
144.                    Xtmi.add(Xmi);
145.                }
146.
147.                // Y[tmi]
148.                IloNumVarMatrix3x3 Ytmi(env,0);
149.                for (int j=0;j<tmax;j++){
150.                    IloNumVarMatrix2x2 Ymi(env,0);
151.                    for (int k=0;k<mmax;k++){
152.                        IloNumVarArray Yi(env,0);
153.                        for (int d=0;d<imax;d++){
154.                            char Path[70];
155.                            sprintf(Path,"Ytmi(t%d,m%d,i%d)",j,k,d);
156.                            IloNumVar Y(env,0,1,ILOBOOL,Path);
157.                            Yi.add(Y);
158.                        }
159.                        Ymi.add(Yi);
160.                    }
```

```
161.                 Ytmi.add(Ymi);
162.             }
163.
164.             // Z[tm]
165.             IloNumVarMatrix2x2 Ztm(env,0);
166.             for (int k=0;k<tmax;k++){
167.                 IloNumVarArray Zm(env,0);
168.                 for (int d=0;d<mmax;d++){
169.                     char Path[70];
170.                     sprintf(Path,"Ztm(t%d,m%d",k,d);
171.                     IloNumVar Z(env,0,1,ILOBOOL,Path);
172.                     Zm.add(Z);
173.                 }
174.                 Ztm.add(Zm);
175.             }
176.
177.             // K[tm]
178.             IloNumVarMatrix2x2 Ktm(env,0);
179.             for (int k=0;k<tmax;k++){
180.                 IloNumVarArray Km(env,0);
181.                 for (int d=0;d<mmax;d++){
182.                     char Path[70];
183.                     sprintf(Path,"Ktm(t%d,m%d",k,d);
184.                     IloNumVar K(env,0,1,ILOBOOL,Path);
185.                     Km.add(K);
186.                 }
187.                 Ktm.add(Km);
188.             }
189.
190.             // c[tm]
191.             IloNumVarMatrix2x2 ctm(env,0);
192.             for (int k=0;k<tmax;k++){
193.                 IloNumVarArray cm(env,0);
194.                 for (int d=0;d<mmax;d++){
195.                     char Path[70];
196.                     sprintf(Path,"ctm(t%d,m%d",k,d);
197.                     IloNumVar c(env,0,1,ILOBOOL,Path);
198.                     cm.add(c);
199.                 }
200.                 ctm.add(cm);
201.             }
202.
203.             /*------------------*
204.              *                  *
205.              *   Constraints    *
206.              *                  *
207.              *------------------*/
208.
209.             // Constraint 2
210.             IloRangeMatrix2x2 Constraint2_2x2(env,0);
211.             for (int x1 = 0; x1 < tmax; x1++){
212.                 IloRangeArray Constraint2_Ar(env,0);
213.                 for (int x2 = 0; x2 < mmax; x2++){
214.                     IloExpr expr(env,0);
215.                     for (int x3 = 0; x3 < imax; x3++){
216.                         expr += Xtmi[x1][x2][x3]+Ytmi[x1][x2][x3];
217.                     }
218.                     expr += Ktm[x1][x2] + Ztm[x1][x2];
219.
220.                     int LB=1, UB=1;
221.                     IloRange Constraint2_(env,LB,expr,UB);
222.                     expr.end();
223.                     model.add(Constraint2_);
224.                     Constraint2_Ar.add(Constraint2_);
225.                 }
```

```
226.                     Constraint2_2x2.add(Constraint2_Ar);
227.                 }
228.
229.                 // Constraint 3
230.                 IloRangeArray Constraint3_Ar(env,0);
231.                 for (int x1 = 0; x1 < tmax; x1++){
232.                     IloExpr expr(env,0);
233.                     for (int x2 = 0; x2 < mmax; x2++){
234.                         for (int x3 = 0; x3 < imax; x3++){
235.                             expr += Xtmi[x1][x2][x3] + Ytmi[x1][x2][x3];
236.                         }
237.                     }
238.
239.                     float LB=0, UB=2;
240.                     IloRange Constraint3_(env,LB,expr,UB);
241.                     expr.end();
242.                     model.add(Constraint3_);
243.                     Constraint3_Ar.add(Constraint3_);
244.                 }
245.
246.                 // Constraint 4-5
247.                 IloRangeArray Constraint4_Ar(env,0);
248.                 for (int x1 = 0; x1 < imax; x1++){
249.                     IloExpr expr(env,0);
250.                     for (int x2 = 0; x2 < tmax; x2++){
251.                         for (int x3 = 0; x3 < mmax; x3++){
252.                             expr += P[x3][x1]*Xtmi[x2][x3][x1];
253.                         }
254.                     }
255.
256.                     float LB=D[x1], UB=Pmin[x1];
257.                     IloRange Constraint4_(env,LB,expr,UB);
258.                     expr.end();
259.                     model.add(Constraint4_);
260.                     Constraint4_Ar.add(Constraint4_);
261.                 }
262.
263.                 // Constraint 6
264.                 IloRangeMatrix3x3 Constraint6_3x3(env,0);
265.                 for (int x1 = 0; x1 < tmax - 1; x1++){
266.                     IloRangeMatrix2x2 Constraint6_2x2(env,0);
267.                     for (int x2 = 0; x2 < mmax; x2++){
268.                         IloRangeArray Constraint6_Ar(env,0);
269.                         for (int x3 = 0; x3 < imax; x3++){
270.                             IloExpr expr(env,0);
271.                             expr = Ytmi[x1][x2][x3]-Xtmi[x1+1][x2][x3]+Xtmi[x1][x2][x3];
272.
273.                             float LB=0, UB=IloInfinity;
274.                             IloRange Constraint6_(env,LB,expr,UB);
275.                             expr.end();
276.                             model.add(Constraint6_);
277.                             Constraint6_Ar.add(Constraint6_);
278.                         }
279.                         Constraint6_2x2.add(Constraint6_Ar);
280.                     }
281.                     Constraint6_3x3.add(Constraint6_2x2);
282.                 }
283.
284.                 // Constraint 7
285.                 IloRangeMatrix3x3 Constraint7_3x3(env,0);
286.                 for (int x1 = 0; x1 < tmax - 1; x1++){
287.                     IloRangeMatrix2x2 Constraint7_2x2(env,0);
288.                     for (int x2 = 0; x2 < mmax; x2++){
289.                         IloRangeArray Constraint7_Ar(env,0);
290.                         for (int x3 = 0; x3 < imax; x3++){
```

```
291.                            IloExpr expr(env,0);
292.                            expr = Xtmi[x1+1][x2][x3] - Xtmi[x1][x2][x3] + 1.1 * (1 -
    Ytmi[x1][x2][x3] );
293.
294.                            float LB=0.1, UB=IloInfinity;
295.                            IloRange Constraint7_(env,LB,expr,UB);
296.                            expr.end();
297.                            model.add(Constraint7_);
298.                            Constraint7_Ar.add(Constraint7_);
299.                        }
300.                        Constraint7_2x2.add(Constraint7_Ar);
301.                    }
302.                    Constraint7_3x3.add(Constraint7_2x2);
303.                }
304.
305.                // Constraint 8
306.                IloRangeMatrix2x2 Constraint8_2x2(env,0);
307.                for (int x1 = 0; x1 < tmax - 1; x1++){
308.                    IloRangeArray Constraint8_Ar(env,0);
309.                    for (int x2 = 0; x2 < mmax; x2++){
310.                        IloExpr expr(env,0);
311.                        for (int x3 = 0; x3 < imax; x3++){
312.                            expr -= Xtmi[x1][x2][x3] + Ytmi[x1][x2][x3] - Xtmi[x1+1][x2][x3] -
    Ytmi[x1+1][x2][x3];
313.                        }
314.
315.                        expr += Ztm[x1+1][x2];
316.                        float LB=0, UB=IloInfinity;
317.                        IloRange Constraint8_(env,LB,expr,UB);
318.                        expr.end();
319.                        model.add(Constraint8_);
320.                        Constraint8_Ar.add(Constraint8_);
321.                    }
322.                    Constraint8_2x2.add(Constraint8_Ar);
323.                }
324.
325.                // Constraint 9
326.                IloRangeMatrix2x2 Constraint9_2x2(env,0);
327.                for (int x1 = 0; x1 < tmax-1; x1++){
328.                    IloRangeArray Constraint9_Ar(env,0);
329.                    for (int x2 = 0; x2 < mmax; x2++){
330.                        IloExpr expr(env,0);
331.                        expr = ctm[x1+1][x2]-Ztm[x1+1][x2]+Ztm[x1][x2];
332.
333.                        float LB=0, UB=1;
334.                        IloRange Constraint9_(env,LB,expr,UB);
335.                        expr.end();
336.                        model.add(Constraint9_);
337.                        Constraint9_Ar.add(Constraint9_);
338.                    }
339.                    Constraint9_2x2.add(Constraint9_Ar);
340.                }
341.
342.                // Constraint 10
343.                IloRangeMatrix2x2 Constraint10_2x2(env,0);
344.                for (int x1 = 0; x1 < tmax-1; x1++){
345.                    IloRangeArray Constraint10_Ar(env,0);
346.                    for (int x2 = 0; x2 < mmax; x2++){
347.                        IloExpr expr(env,0);
348.                        expr = Ztm[x1+1][x2] - Ztm[x1][x2] + 1.1 * ( 1 - ctm[x1+1][x2] );
349.
350.                        float LB=0.1, UB=IloInfinity;
351.                        IloRange Constraint10_(env,LB,expr,UB);
352.                        expr.end();
353.                        model.add(Constraint10_);
```

```
354.                     Constraint10_Ar.add(Constraint10_);
355.                 }
356.                 Constraint10_2x2.add(Constraint10_Ar);
357.             }
358.
359.             // Constraint 11
360.             IloRangeMatrix2x2 Constraint11_2x2(env,0);
361.             for (int x1 = 0; x1 < tmax-1; x1++){
362.                 IloRangeArray Constraint11_Ar(env,0);
363.                 for (int x2 = 0; x2 < mmax; x2++){
364.                     IloExpr expr(env,0);
365.
366.                     expr = Ztm[x1][x2]+Ztm[x1+1][x2]- 2*ctm[x1][x2];
367.                     float LB=0, UB=1;
368.                     IloRange Constraint11_(env,LB,expr,UB);
369.                     expr.end();
370.                     model.add(Constraint11_);
371.                     Constraint11_Ar.add(Constraint11_);
372.                 }
373.                 Constraint11_2x2.add(Constraint11_Ar);
374.             }
375.
376.             // Constraint 12
377.             IloRangeMatrix2x2 Constraint12_2x2(env,0);
378.             for (int x1 = 0; x1 < tmax-23; x1++){
379.                 IloRangeArray Constraint12_Ar(env,0);
380.                 for (int x2 = 0; x2 < mmax; x2++){
381.                     IloExpr expr(env,0);
382.                     for (int x3 = x1; x3 < x1+23; x3++){
383.                         for (int x4 = 0; x4 < imax; x4++){
384.                             expr += Xtmi[x3][x2][x4] + Ytmi[x3][x2][x4];
385.                         }
386.                     }
387.
388.                     int LB=0, UB=22;
389.                     IloRange Constraint12_(env,LB,expr,UB);
390.                     expr.end();
391.                     model.add(Constraint12_);
392.                     Constraint12_Ar.add(Constraint12_);
393.                 }
394.                 Constraint12_2x2.add(Constraint12_Ar);
395.             }
396.
397.             //Constraint13
398.             IloRangeArray Constraint13_Ar(env,0);
399.             for (int x1 = 0; x1 < imax; x1++){
400.                 if ((D[x1]-22*Pi[x1])<=0){
401.                     IloExpr expr(env,0);
402.                     for (int x2 = 0; x2 < tmax; x2++){
403.                         for (int x3 = 0; x3 < mmax; x3++){
404.                             expr += Ytmi[x2][x3][x1];
405.                         }
406.                     }
407.                     float LB=0, UB=1;
408.                     IloRange Constraint13_(env,LB,expr,UB);
409.                     expr.end();
410.                     model.add(Constraint13_);
411.                     Constraint13_Ar.add(Constraint13_);
412.                 }
413.
414.                 else{
415.                     IloExpr expr(env,0);
416.                     for (int x2 = 0; x2 < tmax; x2++){
417.                         for (int x3 = 0; x3 < mmax; x3++){
418.                             expr += Ytmi[x2][x3][x1];
```

```
                        }
                    }
                    float LB=0, UB=3;
                    IloRange Constraint13_(env,LB,expr,UB);
                    expr.end();
                    model.add(Constraint13_);
                    Constraint13_Ar.add(Constraint13_);
                }
            }

            //Constraint 14(free lact)
            IloRangeMatrix2x2 Constraint14_2x2(env,0);
            for (int x1 = 2; x1 < tmax; x1++){
                IloRangeArray Constraint14_Ar(env,0);
                for (int x2 = 0; x2 < mmax-3; x2++){
                    IloExpr expr(env,0);
                    for (int x3 = 0; x3 < imax; x3++){
                        if (x3==0) continue;
                        expr += Xtmi[x1-2][x2][x3];
                    }
                    expr += Xtmi[x1][x2][0];
                    int LB=0 , UB=1;
                    IloRange Constraint14_(env,LB,expr,UB);
                    expr.end();
                    model.add(Constraint14_);
                    Constraint14_Ar.add(Constraint14_);
                }
                Constraint14_2x2.add(Constraint14_Ar);
            }

            //Constraint 15(split)
            IloRangeMatrix2x2 Constraint15_2x2(env,0);
            for (int x1 = 2; x1 < tmax; x1++){
                IloRangeArray Constraint15_Ar(env,0);
                for (int x2 = 0; x2 < mmax-4; x2++){
                    IloExpr expr(env,0);
                    for (int x3 = 0; x3 < imax; x3++){
                        if (x3==6) continue;
                        expr += Xtmi[x1-2][x2][x3];
                    }
                    expr += Xtmi[x1][x2][6];
                    int LB=0 , UB=1;
                    IloRange Constraint15_(env,LB,expr,UB);
                    expr.end();
                    model.add(Constraint15_);
                    Constraint15_Ar.add(Constraint15_);
                }
                Constraint15_2x2.add(Constraint15_Ar);
            }

            //Constraint 16(split)
            IloRangeMatrix2x2 Constraint16_2x2(env,0);
            for (int x1 = 0; x1 < tmax-1; x1++){
                IloRangeArray Constraint16_Ar(env,0);
                for (int x2 = 0; x2 < mmax-4; x2++){
                    IloExpr expr(env,0);
                    for (int x3 = 0; x3 < imax; x3++){
                        if (x3==6) continue;
                        expr += Ytmi[x1+1][x2][x3];
                    }
                    expr += Xtmi[x1][x2][6];
                    int LB=0 , UB=1;
                    IloRange Constraint16_(env,LB,expr,UB);
                    expr.end();
                    model.add(Constraint16_);
```

```
484.                         Constraint16_Ar.add(Constraint16_);
485.                     }
486.                     Constraint16_2x2.add(Constraint16_Ar);
487.                 }
488.
489.                 //Constraint 17(set)
490.                 IloRangeMatrix2x2 Constraint17_2x2(env,0);
491.                 for (int x1 = 2; x1 < tmax; x1++){
492.                     IloRangeArray Constraint17_Ar(env,0);
493.                     for (int x2 = 1; x2 < mmax-3; x2++){
494.                         IloExpr expr(env,0);
495.                         for (int x3 = 0; x3 < imax; x3++){
496.                             if (x3==13) continue;
497.                             if (x3==14) continue;
498.                             if (x3==15) continue;
499.                             if (x3==16) continue;
500.                             expr += Xtmi[x1-2][x2][x3];
501.                         }
502.                         expr += Xtmi[x1][x2][13];
503.                         expr += Xtmi[x1][x2][14];
504.                         expr += Xtmi[x1][x2][15];
505.                         expr += Xtmi[x1][x2][16];
506.                         int LB=0 , UB=1;
507.                         IloRange Constraint17_(env,LB,expr,UB);
508.                         expr.end();
509.                         model.add(Constraint17_);
510.                         Constraint17_Ar.add(Constraint17_);
511.                     }
512.                     Constraint17_2x2.add(Constraint17_Ar);
513.                 }
514.
515.                 //Constraint 18(set)
516.                 IloRangeMatrix2x2 Constraint18_2x2(env,0);
517.                 for (int x1 = 0; x1 < tmax-1; x1++){
518.                     IloRangeArray Constraint18_Ar(env,0);
519.                     for (int x2 = 1; x2 < mmax-3; x2++){
520.                         IloExpr expr(env,0);
521.                         for (int x3 = 0; x3 < imax; x3++){
522.                             if (x3==13) continue;
523.                             if (x3==14) continue;
524.                             if (x3==15) continue;
525.                             if (x3==16) continue;
526.                             expr += Ytmi[x1+1][x2][x3];
527.                         }
528.                         expr += Xtmi[x1][x2][13];
529.                         expr += Xtmi[x1][x2][14];
530.                         expr += Xtmi[x1][x2][15];
531.                         expr += Xtmi[x1][x2][16];
532.                         int LB=0 , UB=1;
533.                         IloRange Constraint18_(env,LB,expr,UB);
534.                         expr.end();
535.                         model.add(Constraint18_);
536.                         Constraint18_Ar.add(Constraint18_);
537.                     }
538.                     Constraint18_2x2.add(Constraint18_Ar);
539.                 }
540.
541.                 //Constraint 19(kefir)
542.                 IloRangeMatrix2x2 Constraint19_2x2(env,0);
543.                 for (int x1 = 0; x1 < tmax - 1; x1++){
544.                     IloRangeArray Constraint19_Ar(env,0);
545.                     for (int x2 = 0; x2 < mmax-3; x2++){
546.                         IloExpr expr(env,0);
547.                         for (int x3 = 0; x3 < imax; x3++){
548.                             if (x3==12) continue;
```

```
549.                            expr += Ytmi[x1+1][x2][x3];
550.                        }
551.                        expr += Xtmi[x1][x2][12];
552.                        int LB=0 , UB=1;
553.                        IloRange Constraint19_(env,LB,expr,UB);
554.                        expr.end();
555.                        model.add(Constraint19_);
556.                        Constraint19_Ar.add(Constraint19_);
557.                    }
558.                    Constraint19_2x2.add(Constraint19_Ar);
559.                }
560.
561.                //Constraint 20(mpiskoto)
562.                IloRangeMatrix2x2 Constraint20_2x2(env,0);
563.                for (int x1 = 0; x1 < tmax - 1; x1++){
564.                    IloRangeArray Constraint20_Ar(env,0);
565.                    for (int x2 = 3; x2 < mmax-1; x2++){
566.                        IloExpr expr(env,0);
567.                        for (int x3 = 0; x3 < imax; x3++){
568.                            if (x3==19) continue;
569.                            expr += Ytmi[x1+1][x2][x3];
570.                        }
571.                        expr += Xtmi[x1][x2][19];
572.                        int LB=0 , UB=1;
573.                        IloRange Constraint20_(env,LB,expr,UB);
574.                        expr.end();
575.                        model.add(Constraint20_);
576.                        Constraint20_Ar.add(Constraint20_);
577.                    }
578.                    Constraint20_2x2.add(Constraint20_Ar);
579.                }
580.
581.                //Constraint 21(mix)
582.                IloRangeMatrix2x2 Constraint21_2x2(env,0);
583.                for (int x1 = 0; x1 < tmax - 1; x1++){
584.                    IloRangeArray Constraint21_Ar(env,0);
585.                    for (int x2 = 0; x2 < mmax-3; x2++){
586.                        IloExpr expr(env,0);
587.                        for (int x3 = 0; x3 < imax; x3++){
588.                            if (x3==10) continue;
589.                            expr += Ytmi[x1+1][x2][x3];
590.                        }
591.                        expr += Xtmi[x1][x2][10];
592.                        int LB=0 , UB=1;
593.                        IloRange Constraint21_(env,LB,expr,UB);
594.                        expr.end();
595.                        model.add(Constraint21_);
596.                        Constraint21_Ar.add(Constraint21_);
597.                    }
598.                    Constraint21_2x2.add(Constraint21_Ar);
599.                }
600.
601.                // Constraint 22
602.                IloRangeMatrix2x2 Constraint22_2x2(env,0);
603.                for (int x1 = 0; x1 < mmax; x1++){
604.                    IloRangeArray Constraint22_Ar(env,0);
605.                    for (int x2 = 0; x2 < imax; x2++){
606.                        IloExpr expr(env,0);
607.                        expr = Xtmi[0][x1][x2];
608.
609.                        int LB=0, UB=0;
610.                        IloRange Constraint22_(env,LB,expr,UB);
611.                        expr.end();
612.                        model.add(Constraint22_);
613.                        Constraint22_Ar.add(Constraint22_);
```

```
614.                     }
615.                     Constraint22_2x2.add(Constraint22_Ar);
616.                 }
617.
618.                 // Constraint 23
619.                 IloRangeMatrix2x2 Constraint23_2x2(env,0);
620.                 for (int x1 = 0; x1 < mmax; x1++){
621.                     IloRangeArray Constraint23_Ar(env,0);
622.                     for (int x2 = 0; x2 < imax; x2++){
623.                         IloExpr expr(env,0);
624.                         expr = Xtmi[166][x1][x2]+Ytmi[166][x1][x2]+Xtmi[167][x1][x2]+Ytmi[167][x1][x2]
    ;
625.
626.                         int LB=0, UB=0;
627.                         IloRange Constraint23_(env,LB,expr,UB);
628.                         expr.end();
629.                         model.add(Constraint23_);
630.                         Constraint23_Ar.add(Constraint23_);
631.                     }
632.                     Constraint23_2x2.add(Constraint23_Ar);
633.                 }
634.
635.                 // Constraint40
636.                 IloExpr expr(env,0);
637.                 for (int x1 = 0; x1 < tmax; x1++){
638.                     for(int x2 = 0; x2< mmax; x2++){
639.                         expr += Ztm[x1][x2];
640.                     }
641.                 }
642.                 int LB=0 , UB=32;
643.                 IloRange Constraint40_(env,LB,expr,UB);
644.                 expr.end();
645.                 model.add(Constraint40_);
646.
647.                 /*------------------*
648.                  *                  *
649.                  *  Objective Func  *
650.                  *                  *
651.                  *------------------*/
652.
653.                 IloExpr expr_obj(env);
654.                 for (int x1=0; x1<tmax; x1++){
655.                     for (int x2=0; x2<mmax; x2++){
656.                         for (int x3=0; x3<imax; x3++){
657.                             expr_obj += Xtmi[x1][x2][x3]+Ytmi[x1][x2][x3];
658.                         }
659.                     }
660.                 }
661.                 expr_obj *= Cl;
662.                 for (int x1=0; x1<tmax; x1++){
663.                     for (int x2=0; x2<mmax; x2++){
664.                         for (int x3=0; x3<imax; x3++){
665.                             expr_obj += Ytmi[x1][x2][x3] * Cs[x3];
666.                         }
667.                     }
668.                 }
669.
670.                 for (int x1=0; x1<tmax; x1++){
671.                     for (int x2=0; x2<mmax; x2++){
672.                         expr_obj += Ztm[x1][x2] * Cc[x2];
673.                     }
674.                 }
675.                 model.add(IloMinimize(env, expr_obj));
676.                 expr_obj.end();
677.
```

```
678.                /*------------------*
679.                 *                  *
680.                 *     Solution     *
681.                 *                  *
682.                 *------------------*/
683.
684.                // Solve
685.                cplex.extract(model);
686.                cplex.exportModel("sol.lp");
687.                // Print Results
688.                if (!cplex.solve ()){
689.                    env.error()<<"Faild to optimize LP."<<endl;
690.                    throw(-1);
691.                }
692.
693.                env.out()<<"Solution status = " <<cplex.getStatus()<<endl;
694.                env.out()<<"Solution value = " <<cplex.getObjValue()<<endl;
695.
696.
697.                cplex.solve();
698.
699.                ofstream myfile;
700.                myfile.open("output.txt");
701.                for (int j=0;j<tmax;j++){
702.                    for (int k=0;k<mmax;k++){
703.                        int g = cplex.getValue(Ktm[j][k]);
704.                        if(g!=0) {
705.                            std::ostringstream oss;
706.                            oss <<"Ktm"<<"("<<j<<","<<k<<")"<<"="<<g<<endl;
707.                            std::string var = oss.str();
708.                            myfile << var;
709.                        }
710.                        g = cplex.getValue(Ztm[j][k]);
711.                        if(g!=0) {
712.                            std::ostringstream oss;
713.                            oss <<"Ztm"<<"("<<j<<","<<k<<")"<<"="<<g<<endl;
714.                            std::string var = oss.str();
715.                            myfile << var;
716.                        }
717.                        for (int d=0;d<imax;d++){
718.                            g = cplex.getValue(Xtmi[j][k][d]);
719.                            if(g!=0) {
720.                                std::ostringstream oss;
721.                                oss <<"Xtmi"<<"("<<j<<","<<k<<","<<d<<")"<<"="<<g<<endl;
722.                                std::string var = oss.str();
723.                                myfile << var;
724.                            }
725.                            g = cplex.getValue(Ytmi[j][k][d]);
726.                            if(g!=0) {
727.                                std::ostringstream oss;
728.                                oss <<"Ytmi"<<"("<<j<<","<<k<<","<<d<<")"<<"="<<g<<endl;
729.                                std::string var = oss.str();
730.                                myfile << var;
731.                            }
732.                        }
733.                    }
734.                }
735.                myfile.close();
736.
737.
738.            }
739.            catch ( IloException& e){
740.                cerr <<"concert exception caught:"<<e<<endl;
741.            }
742.            catch (...){
```

```
743.                 cerr <<"Unknown exception caught"<<endl;
744.             }
745.             // End env
746.             env.end();
747.
748.             system("pause");
749.             return 0;
750.         }
```

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Doganis, P.; Sarimveis, H.s;. (2008). Optimal production scheduling for the dairy industry. (159), pp. 315-331.

[2] Entrup, M; Gunther, H.O.; Van Beek, P.; Seiler, T. (2005). Mixed integer linear programing approaches to shelf life integrated planning and scheduling in yogurt production. (43), pp. 5071-5100.

[3] Kopanos, G., Puigjaner, L., &Georgiadis, M. (2010). Optimal Production Scheduling and Lot sizing in Dairy Plants: The yogurt production line. Ind.Eng.Chem. (49), pp. 701-718.

[4] Doganis, P.; Sarimveis, H.s; Bafas, G.e. (2005). Optimal production scheduling for dairy industries based on a neural network saled forecasting model. In V. D. France (Ed.), IMACS world congress. Lille.

[5]Marinelli, F; Nenni, M.; Sforza, A. (2007). Capacitated lot sizing and scheduling with parallel machines and shared buffers. A case study in a packaging company. (150), pp. 177-192.

[6] Lee, K., Park, H. I., & Lee, I. (2001). A novel non uniform discrete time formulation for short-term scheduling of batch and continuous processes. Industrial and Engineering Chemistry Research, 40, 4902– 4911.

[7] Lin, X., & Floudas, C. A. (2001). Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. Computers and Chemical Engineering, 25, 665–674.

[8] Lin, X., Floudas, C. A., Modi, S., & Juhasz, N. M. (2002). Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. Industrial and Engineering Chemistry Research, 41, 3884–3906.

[9] Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling. Part 1. Multipurpose batch processes. Industrial and Engineering Chemistry Research, 37, 4341– 4374.

[10] Kozanidis G, Liberopoulos G, Pitsilkas C (2010) Flight and maintenance planning of military aircraft for maximum fleet availability. Military Operations Research 15(1):53-73.

[11] Liberopoulos G, Pandelis DG, Hatzikonstantinou O (2013) The stochastic economic lot sizing problem for non-stop multi-grade production with sequence-restricted setup changeovers. Annals of Operations Research 209(1):179-205.

[12] Mockus, L., & Reklaitis, G. V. (1999). Continuous time representation approach to batch and continuous process scheduling. Part 1. MINLP formulation, . Industrial and Engineering Chemistry Research, 38, 197– 210.

[13]George L. Nemhauser, Laurence A. Wolsey. (1998). Integer and combinatorial optimization, John Wiley & Sons, Inc.

[14] William J. Cook, William H. Cunningham, William R. Pulleyblank, Alexander Schrijver. (1997). Combinatorial Optimization,Wiley-Interscience.

[15] Alexander Schrijver. (1998). Theory of Linear and Integer Programming, John Wiley & Sons, Inc.

[16] Laurence A. Wolsey. (1998).Integer Programming, John Wiley & Sons, Inc.

[17] Bowman, E. H. (1959). The schedule-sequencing problem. Operations Research, 7, 621–624.

[18] Hui, C., Gupta, A., & van der Meulen, H. A. J. (2000). A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers and Chemical Engineering, 24, 2705–2717.

[19] Eren, T. (2007). A multicriteria flowshop scheduling problem with setup times. Journal of Materials Processing Technology, 186, 60-65.

[20] Lamba, N., & Karimi, L. A. (2002a). Scheduling parallel production lines with resource constraints. Part 1. Model formulation. Industrial and Engineering Chemistry Research, 41, 779–789.

[21] Aydilek, H., & Allahverdi, A. (2006). Two-machine flowshop scheduling problem with bounded processing times to minimize total completion time. International Journal of Production Economics, 103 (1), 286-400.

[22] Κολέτσος Ι., Στογιάννης Δ. (2012). Εισαγωγή στην Επιχειρησιακή Έρευνα, εκδ. Συμεών.

[23] Τσιούτρα Σ. (2013). Βέλτιστος προγραμματισμός παραγωγής σε γαλακτοβιομηχανία για την περίπτωση του γιαουρτιού, Πανεπιστήμιο Θεσσαλίας. Πολυτεχνική Σχολή. Τμήμα Μηχανολόγων Μηχανικών.