UNIVERSITY OF GAZİANTEP

GRADUATE SCHOOL OF

NATURAL & APPLIED SCIENCES

DYNAMIC MANUFACTURING CELL FORMATION THROUGH MARKET ORIENTED PROGRAMMING

Ph.D THESIS

IN

INDUSTRIAL ENGINEERING

BY

LATİFE GÖRKEMLİ

JANUARY 2014

Dynamic Manufacturing Cell Formation through Market Oriented Programming

Ph.D Thesis

in

Industrial Engineering

University of Gaziantep

Supervisor

Prof. Dr. Adil BAYKASOĞLU

by

Latife GÖRKEMLİ

January 2014

REPUBLIC OF TURKEY
UNIVERSITY OF GAZİANTEP
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
NAME OF THE DEPARTMENT

Name of the thesis: Dynamic manufacturing cell formation through market oriented programming
Name of the student: Latife Görkemli
Exam date: 17.01.2014

Approval of the Graduate School of Natural and Applied Sciences

Assoc. Prof. Dr. Metin BEDİR
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Türkay DERELİ
Head of Department

This is to certify that we have read this thesis and that in our consensus/majority opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Adil BAYKASOĞLU
Supervisor

Examining Committee Members	Signature
Prof. Dr. Adil BAYKASOĞLU	………………
Prof. Dr. Hadi GÖKÇEN	………………
Prof. Dr. Rızvan EROL	………………
Prof. Dr. Türkay DERELİ	………………
Assoc. Prof. Dr. Kürşad AĞPAK	………………

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Latife GÖRKEMLİ

ABSTRACT

DYNAMIC MANUFACTURING CELL FORMATION THROUGH MARKET ORIENTED PROGRAMMING

GÖRKEMLİ, Latife
Ph.D. in Industrial Eng.
Supervisor: Prof. Dr. Adil BAYKASOĞLU
January 2014
100 page

In today's competitive environment, cellular manufacturing is a promising approach providing both the flexibility of job shops and efficiency of flow lines. However, one of the drawbacks of cellular manufacturing and its algorithms is their inability to handle dynamic events, especially dynamic changes in part spectrum. Although there are various efforts in the literature, researchers still could not overcome this problem efficiently. Since handling dynamism with traditional methods is nearly impossible, and the reconfiguration of the cells according to each change is difficult and costly especially in volatile manufacturing systems. In this context, agent based modelling provides opportunities to model dynamism and to obtain efficient solutions. Since it has ability to track and evaluate the real time information if it is implemented successfully. On the other side, virtual cell formation concept provides the opportunity to create manufacturing cells without the reconfiguration. In this thesis study, it is mainly focussed on these modelling approaches to develop a dynamic cellular manufacturing system. And an integrated novel agent based virtual cellular manufacturing approach is developed. The proposed approach enables to realize part family formation, virtual cell formation, and scheduling simultaneously while considering dynamic part demand arrivals. The results are discussed and it is shown that the proposed approach is very effective.

Key Words: Agent based modelling, virtual cellular manufacturing, dynamism.

ÖZET

PİYASA ODAKLI PROGRAMLAMA İLE DİNAMİK ÜRETİM HÜCRELERİNİN OLUŞTURULMASI

GÖRKEMLİ, Latife
Doktora Tezi, Endüstri Müh. Bölümü
Tez Yöneticisi: Prof. Dr. Adil BAYKASOĞLU
Ocak 2014
100 sayfa

Bugünün rekabetçi imalat sektöründe hücresel imalat yöntemi, atölye tipi üretimin esnekliğini ve seri üretimin etkinliğini içinde bulundurmasıyla umut vaad eden bir yaklaşımdır. Fakat hücresel imalat ve algoritmalarının en önemli eksikliklerinden birisi özellikle parça taleplerinde meydana gelen dinamik değişiklikleri olmak üzere dinamik olarak meydana gelen olayları modellemedeki yetersizliğidir. Literatürde bu konuda birçok çalışma yapılmasına rağmen hala bu problemin üstesinden etkin bir şekilde gelinememiştir. Çünkü geleneksel yaklaşımlar ile dinamikliği modellemek neredeyse imkansızdır ve meydana gelen her bir değişikliğe göre geleneksel olarak hücreleri yeniden oluşturmak zor ve maliyetli bir iştir. Bu bağlamda, etmen tabanlı modelleme yaklaşımı, dinamikliği modelleme ve etkin sonuçlar elde edilmesinde bir çok avantaja sahiptir. Çünkü etmen tabanlı modellemenin, eğer doğru uygulanırsa, zaman içerisinde meydana gelen değişiklikleri izleme ve değerlendirme kabiliyeti vardır. Diğer taraftan, sanal hücresel imalat yöntemi ile ise imalat hücreleri sanal olarak oluşturulduğundan hücrelerin fiziksel olarak yeniden yapılandırılmasına gerek yoktur. Bu tez çalışmasında hücresel imalat sisteminde dinamikliği modellemede önemli fırsatlar sunan bu yöntemler dikkate alınarak dinamik etmen tabanlı bir sistem geliştirilmiştir. Önerilen sistem ile dinamik olarak gelen parça talepleri dikkate alınarak parça ailesi oluşturma, sanal imalat hücresi oluşturma ve çizelgeleme işlemleri eş zamanlı olarak gerçekleştirilmektedir. Elde edilen sonuçlar değerlendirilmiş ve önerilen sistemin etkinliği ortaya koyulmuştur.

Anahtar Kelimeler: Etmen tabanlı modelleme, sanal hücresel imalat, dinamizm.

To My Family

ACKNOWLEDGEMENTS

I would like to express my thanks and appreciation to my supervisor Prof. Dr. Adil BAYKASOĞLU for his invaluable guidance, support and encouragement throughout the thesis study.

I thank to Prof. Dr. Türkay DERELİ for his valuable support. I also would like to thank the other committee member of my thesis Doç. Dr. Kürşad AĞPAK.

I also thank to my friend Zeynep Didem UNUTMAZ DURMUŞOĞLU for her valuable concern and help.

I am grateful to TUBİTAK BİDEB for their Ph.D. scholarship program supported me throughout the Ph. D. studies.

My special thanks for my family; my father, mother, sisters Beyza and Burcu, and brother Nuhkan for their invaluable support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES Page

LIST OF FIGURES

LIST OF ABREVIATIONS

x_{ij}	x_{ij}=1 if part i belongs to part family j, and x_{ij}=0 otherwise
n	number of parts
p	number of part families
m_j	the size of part family j
d_{ij}	Distance between part i and part j
D_{pk}	Distance between part p and part family k
$C_{k,a}$	Center of part family k for attribute a
G_a	Global center of system for attribute a
e	total number of 1's in the matrix
e_0	total number of exceptional elements
e_v	total number of voids
REs	Resource elements
ADS_{if}	Average dissimilarity between part i and part family f
DS_{ij}	Overall dissimilarity level between part i and part j
PDS_{ij}	Part dissimilarity based on commonality of machine requirements between part i and part j
SDS_{ij}	Part dissimilarity considering processing sequences of part i and part j
P_i	Operation sequences of part i
M	n*m matrix
w_i	weight on dissimilarity index i
RT_i	Release time of ith part
DDE_i	Due date estimation of ith part
FTE_j	flow time estimation of ith part

ABVCM-1	Version 1 of agent based virtual cellular manufacturing methodology
ABVCM-2	Version 2 of agent based virtual cellular manufacturing methodology
t	threshold
cce	cell capacity estimation parameter
fte	flow time estimation parameter
dde	due date estimation parameter
EDD	Earliest due date
SPT	Shortest processing time
MLB	Minimum load based
MQLB	Minimum queue length based
EXPO	Exponential distribution

CHAPTER 1

INTRODUCTION

Efficiency and flexibility are the two important keywords of successful manufacturing. A manufacturing system which keeps both of them is desired in most of the manufacturing areas. However, handling such a manufacturing system is not enough in today's competitive world. As most environmental factors change rapidly in this global world, customer demands change rapidly either. Nowadays, there is a need for a manufacturing system which keeps efficiency and flexibility while tracking and evaluating the real time information.

As known, flow lines and job shops have come into prominence having high efficiency and flexibility, respectively. Thus, cellular manufacturing systems which contain features of flow lines and job shops have been studied by researchers for many years. Besides its several advantages, one of the most important drawbacks of cellular manufacturing is that in volatile manufacturing environments cellular manufacturing systems become inapplicable because of the difficulty and cost of reconfiguration. Despite this disadvantage, researchers have been studying how to model dynamism in part demand in cellular manufacturing appropriately. This is because modelling dynamism is very important to obtain meaningful solutions to real world problems. In this context, researchers have mainly focused on modelling changes in demand via multi period cellular manufacturing approaches, such as those proposed by Turker (1993), Chen (1998), Balakrishnan and Cheng (2005), Safael et al. (2007), Muruganandam et al. (2008), Ah kioon et al. (2009), Das and Abdul-Kader (2011), Ghotboddini et al. (2011), and Saxena and Jain (2011). However, in multi period cellular manufacturing, it is assumed that a multi period plan is possible (Balakrishnan and Cheng, 2007). Although changes in demand are modelled by multi period approaches, having knowledge of multi period plans makes the problem static. Therefore, the developed algorithms which work with the assumption of known multi period plans do not have the ability to model dynamic changes in part demands efficiently. Already, in most production systems, a sudden part demand or unexpected demand cancellation causes problems.

Basically, in cellular manufacturing parts and machines are grouped according to their features, and assignments of part families to the machine cells are realised. In volatile manufacturing environments, even dynamism is modelled, as applicability is nearly impossible via traditional cellular manufacturing methods. This is because it is very difficult and costly to reconfigure the manufacturing cells according to each dynamic change in the environment. In the early 1980s, the virtual manufacturing cell concept was introduced by McLean et al. (1982). Mainly, a virtual cell differs from a traditional cell in terms of configuration. The virtual cell is a logical group of machines, thus by using this concept; cellular manufacturing approach can be applied to the manufacturing systems without reconfiguration of machines. But as known, traditional cells are physical groups of machines.

Although virtual cellular manufacturing systems give the opportunity to handle dynamic part changes in part demand spectrum, unfortunately there is no integrated study considering both the main phases of cellular manufacturing and dynamism on part demand arrivals in the manufacturing systems efficiently. In this context, there is big a gap in the literature on this important topic. Modelling cell formation problem with this kind of dynamism and gathering efficient solutions with classical approaches is extremely difficult. This is because these classical approaches create solutions to the definite states of the system, and they are incapable of adapting the changing environmental conditions (Karageorgos et al., 2003; Baykasoglu et al., 2011; Erol et al., 2012). We proposed an agent based modelling approach for dynamic virtual cellular manufacturing systems.

Agent oriented computing provides a marvellous opportunity to handle dynamic problems and to provide effective solutions, if carefully and intelligently implemented. In this study, besides agent based modelling, other modelling concepts and methods which support modelling desired operational issues and dynamism efficiently are brought together such as resource elements, capability based distributed layout, and market oriented programming. Thus, an integrated novel agent based virtual cellular manufacturing methodology is developed. The proposed approach enables to realize part family formation, virtual cell formation, and scheduling simultaneously while considering dynamic part demand arrivals.

The proposed approach is realized on AnyLogic^R simulation platform which presents several advantages while modelling dynamism and agent based systems. In relation to the increasing usage of agent based modelling and simulation approaches to the problems, the number of software tools supporting these methodologies has also increased. AnyLogic^R is one of these software tools, providing different modelling paradigms in any combination such as discrete event, system dynamics, continuous and dynamic system and agent based modelling (As of May 8, 2013, AnyLogic^R mentioned on its website http://www.anylogic.com/overview). One can define most behaviours of the agents using AnyLogic^R. AnyLogic^R simplifies development of agent based models with its designed patterns such as model architecture, agent synchronization, animation, agent connections and communication, dynamic creation and destruction of agents (As of May 8, 2013, AnyLogic^R mentioned on its website http://www.anylogic.com/agent-based-modeling).

In the thesis study, an approach which aims to handle operational issues of manufacturing system and dynamism in part demand arrivals is presented. Firstly we focused on part family formation phase of cellular manufacturing under dynamic part demand changes. In this process, studies are concentrated in conceptual level. Here, we aimed to investigate whether the developed agent based part family formation algorithm can find efficient solutions to the problems while tracking and evaluating the changes. Therefore, the results are exciting and show that the proposed algorithm has an ability to follow optimal solutions in dynamic circumstances. With this motivation, we developed dynamic agent based virtual cellular manufacturing approach. The thesis study is organized as in the following.

1.1 Thesis Organization

Chapter 2: the literature review is presented in this chapter. Studies on cellular manufacturing and virtual cellular manufacturing are examined. And dynamic clustering methods are investigated in this chapter. Also agent based modeling and the properties are presented.

Chapter 3: the developed algorithm for dynamic part family formation algorithm in conceptual level is presented. We attempt to compare the performance of the present algorithm on static test problems by dynamically introducing parts in the literature datasets to our algorithm. Many results have been presented on these static datasets by utilizing several heuristics, meta-heuristics and optimization based algorithms. It is shown that the proposed algorithm has the ability to produce very good solutions which are comparable to the best known results.

Chapter 4: an overview of the proposed dynamic agent based virtual cellular manufacturing approach is presented in this chapter. Also resource elements approach, capability based distributed layout, and market oriented programming methods are explained.

Chapter 5: the details of the proposed agent based dynamic virtual cellular manufacturing approach are presented. The properties of the defined agents, the steps of the part family formation and virtual cell formation and scheduling algorithms are given in this chapter.

Chapter 6: the parameters and their effects on the performance of the proposed algorithm and the performance of the algorithm are analyzed in this chapter. The analyses are divided into three parts. In the first part the parameters which directly affect the performance of part family formation are examined and their effect on the performance is discussed. In the second the scheduling rules are investigated. And in the third part, performance of the proposed algorithm compared with the results of the manufacturing system which mainly has the same properties but works as functional job shop.

Chapter 7: the summary of the study and the conclusions are given in the last chapter. Also some aspects for the future work are discussed.

CHAPTER 2

LITERATURE REVIEW

Cellular manufacturing comes into prominence with several advantages such as reduction in production lead time, reduction in labor, reduction in set up time, and improvement in scheduling and planning (Baykasoglu and Gindy, 2000; Baykasoglu et al., 2001; Baykasoglu, 2004). Cell formation process which includes part clustering and machine assignment to these clusters, is one of the most important phases of cellular manufacturing (Baykasoglu et al., 1998a; Keeling et al., 2007). In the literature, much work has been undertaken in order to gather effective solutions to the cell formation problem using different constraints and objectives. There are several review papers which provide extensive classification and evaluation of these approaches. Selim et al. (1998) provided a mathematical programming formulation for the cell formation problem. They also presented a methodology based classification on the cell formation problem. Papaioannou and Wilson (2010) presented a literature review of the cell formation problem, concentrating on formulations proposed between 1997 and 2008. In their paper a comparison and an evaluation of the methodologies were performed and a number of conclusions deduced. Yin and Yasuda (2006) presented an overview and discussion on similarity coefficients developed for cell formation. They developed taxonomy to explain the definition and usage of the similarity coefficients. Sarker (2001) presented a review on categorization and generalization of the measures developed for the determination of the goodness of machine-part groups in cellular manufacturing systems. They also proposed a new grouping efficiency measure. Balakrishnan and Cheng (2007) reviewed the studies performed to address issues related to multi-period planning horizons with demand and resource uncertainties in cellular manufacturing. They stated that most traditional cell formation approaches ignore any changes in demand over time; it is assumed that part demand stays constant over long periods of time.

However, changes in demand occur because of the product redesign and uncertainties due to volume variation, part mix variation, and resource unreliability (Balakrishnan and Cheng, 2007). Turker (1993), Chen (1998), Balakrishnan and Cheng (2005), Safael et al. (2007), Muruganandam et al. (2008), Ah kioon et al. (2009), Rezazadeh et al. (2011), Ghotboddini et al. (2011), Saxena and Jain (2011) and Das and Abdul-Kader (2011) presented studies considering multiple time periods in cellular manufacturing.

In today's business environment, part demand and mix can change rapidly and unexpectedly. Thus, a cell formation methodology needs to address these issues (Balakrishnan and Cheng, 2007; Baykasoglu et al., 1998b; Saad et al., 2002a). However, as seen in the literature, changes in production environment related to dynamic changes in part demand are predominately evaluated in multi-period cell formation approaches. Balakrishnan and Cheng (2007) mentioned that in multi period cellular manufacturing systems, it is assumed that a multi period plan is possible. Consequently, these multi period cell formation approaches do not address fully dynamic problems. In fact, related to this context, the solved problems are static as no change occurs dynamically in part demand changes, they are assumed to be known beforehand for each period. According to the literature review, there is only one study which considers dynamic part arrivals in the part family formation problem in cellular manufacturing without the assumption of known part type at the beginning of the problem solution. This study is presented by Ben-Arieh and Sreenivasan (1999). They proposed a methodology which allows parts to be grouped as they arrive. Also, the existing parts can change their part families without the need to solve the part family formation problem from the beginning. Their algorithm can be considered as a distributed, dynamic and negotiation based method.

In actual fact, part family formation process is clustering parts considering some of their properties. So, dynamic clustering methodologies in the literature are examined although they are not applied to the part family formation problem. Khalilian and Mustapha (2010) gave several example areas that require dynamic processing: network monitoring, calling records, sensor monitoring, stock exchange, power supply and manufacturing, examining the spread of illnesses etc. Clustering these streaming data which is not completely known at the beginning of the clustering process is studied as data stream clustering in the literature (Fournier et al., 2007). Khalilian and Mustapha (2010) stated two main problems focused on the data stream clustering in the literature as in: 1) visiting data once because of the insufficiency of data storage capacity, and 2) evolution of streaming data and concept change during time. Charu et al. (2003) presented an approach which contains two components as online micro clustering and offline macro clustering. During online component, detailed summary statistics are stored periodically. This summary statistics is used by the offline component. The method called ClueStream framework also provides exploration of the evolution of the clusters over different time periods. Fournier et al. (2007) proposed a multi-agent algorithm for dynamic clustering. The proposed approach combines an ants algorithm with agent theory and executes these algorithms simultaneously. Kiselev and Alhajj (2008) presented an adaptive multi-agent approach to continuous online clustering of streaming data which is sensitive to environmental variations. In their approach market-based negotiation is used to model the unsupervised clustering as a dynamic distributed allocation problem. Sandhir and Kumar (2010) mentioned that many real world applications require online analysis of streaming data, and they proposed an algorithm which is modification of the fuzzy c-means clustering technique. The proposed algorithm allows clusters to be adaptively updated as data points keep streaming in. Lee et al. (2011) developed a framework for online anomaly detection. In the study, a self organizing map (SOM) is combined with K-means clustering. According to the proposed dynamic algorithm, an initial model is constructed, and then, depending on the online data the model, evolves gradually. Among the clustering algorithms, Ben-Arieh and Sreenivan’s (1999) algorithm comes into prominence having all the following advantages: handling dynamics effectively without the need to solve the clustering problem from the beginning, no need to determine parameters such as the initial number of clusters, threshold value, and having non-complex computation. It is an agent based approach.

The success of agent based approaches on modelling dynamic systems are already known in the literature and nowadays researchers focus on agent based approaches to solve complex dynamic problems. Davidsson et al. (2003) analyzed the strengths and weaknesses of the agent based approaches. According to their evaluation, agent based approaches are preferable in sequential situations: the domain of problem is large and modular in nature, the probability of failure is high, the time-scale of the domain is short, the structure of domain changes frequently, and there is sensitive information that should be kept locally. If the properties of the dynamic part family formation problem is taken into account, agent based modelling is also a promising approach to this dynamic problem. Since, in dynamic part family formation problem, dynamic arrivals and cancellations are possible in the system and the clustering concept can change depending on the data.

Agent based modelling and simulation is a powerful approach for analyzing and modelling complex systems by making use of autonomous, interacting agents (Macal and North, 2009; Garro and Russo, 2010). Properties of agents are defined as follows (Macal and North, 2009):

- Agents are autonomous and self-directed individuals. They can perform autonomously in their environment and with other agents.
- Agents are modular or self-contained discrete individuals with several attributes, behaviours, and decision making ability.
- Agents are social, interacting individuals. They have protocols which describe communication and information sharing with other agents.
- Agents may have goals to which they evaluate the outcomes of behaviours continuously. And they modify their behaviours in respect to this benchmark.
- Agents may learn and adjust their behaviours based on the experiences.

Borshchev and Flippov (2004) presented a practical reference on agent based modelling. They emphasized that agent based models are decentralized, that is, one defines behaviour of an agent at individual level, and the global behaviour obtained from all the system individuals. They mentioned that since an agent based model enables to handle more complex systems and dynamics, it is more general and powerful. And also maintaining the agent based models is easier. One can construct models in the absence of knowledge about the global behaviour through agent based modelling (Borshchev and Flippov, 2004).

Basically, in cellular manufacturing parts and machines are grouped according to their features, and assignments of part families to the machine cells are realised. In highly volatile manufacturing environments, even dynamism is modelled, as applicability is nearly impossible via traditional cellular manufacturing methods. This is because, it is very difficult and costly to reconfigure the manufacturing cells according to each dynamic change in the environment. In the early 1980s, the virtual manufacturing cell concept was introduced by McLean et al. (1982). Mainly, a virtual cell differs from a traditional cell in terms of configuration. A detailed explanation on virtual cellular manufacturing systems can be found in the study of Drolet (1989), who developed algorithms for the scheduling of these systems. The virtual cell is a logical group of machines; thus, by using this concept, the cellular manufacturing approach can be applied to the manufacturing systems without reconfiguration of machines. However, as known, traditional cells are physical groups of machines. This main difference makes the virtual manufacturing approach promising and there has been an increasing interest in the literature on this topic. However, according to the literature review, there is no integrated study which considers both the main phases of cellular manufacturing and dynamism on part demand arrivals in manufacturing systems efficiently. Some studies focused on configuring virtual cells by considering the different constraints and features of manufacturing systems, but they were not sufficiently able to handle dynamism on part demand arrivals. In this sense, we can classify these studies into two subclasses.

In the first subclass, studies with the assumption of known part demand (all the part demands are available) at the beginning of problem solution are examined. Sarker and Li (2001) proposed a method which adopts the double-sweep algorithm for the k-shortest path problem for virtual cell formation. They also presented a heuristic to schedule the virtual cells when there are multiple job orders. Ko and Egbelu (2003) proposed a virtual cell formation procedure by considering a machine sharing procedure. In the study, machine cell formation was realised by using the routings of parts in the part mix. According to the study, if the new production order differs from the product mix, as before, new virtual cells are formed by the proposed algorithm. Mak et al. (2005) developed a mathematical model and an age-based genetic algorithm for virtual manufacturing cell formation and scheduling. Mak et al. (2007) presented a methodology which consists of a mathematical model that describes the characteristics of a virtual cellular manufacturing system and an ant colony optimisation algorithm for manufacturing cell formation and production scheduling. Kesen et al. (2010a) developed a multi objective mixed integer programming formulation for the scheduling of virtual cells. Kesen et al. (2010b) presented a multi objective mixed integer programming formulation and a genetic algorithm based heuristic approach for job scheduling in virtual manufacturing cells. This study was generalised by Kesen and Güngör (2012) allowing a lot streaming strategy. Khilwani et al. (2011) proposed a mathematical model and a solution procedure for virtual cellular manufacturing. According to the proposed approach, firstly machines are assigned to the cells, then parts are assigned to the cells with maximum similarity index, and then a search algorithm is executed in order to find the best configuration of virtual cells. Hamedi et al. (2012) presented a multi objective mathematical model with a goal programming approach to form capability based virtual cellular manufacturing systems. In the model, worker constraints are also considered. They solved the proposed model through a multi objective tabu search algorithm.

In the second class, there are studies considering multi time periods with the assumption of known part demand at the beginning of each time period. An integrated framework which is mainly based on multiple objective simulation optimisation was proposed by Saad et al. (2002b) for the reconfiguration of cellular manufacturing systems using virtual cells. They stated that part spectrum and demand are not stable and change from one production horizon to another. The decisions related to reconfiguration were made considering the demand of each production horizon by the proposed framework. Mahdavi et al. (2009) developed a mathematical model for manufacturing cell formation and production planning in virtual cellular manufacturing systems with worker flexibility by considering a multi period planning horizon. Mahdavi et al. (2011) proposed a fuzzy goal programming based method to solve a multi objective mathematical model of virtual cell formation and planning which considered worker flexibility. Murali et al. (2010) presented an approach which is based on artificial neural networks. They assigned workers into virtual cells using artificial neural networks by considering different time periods. Murali (2012) expanded their previous study by applying the learning vector quantisation approach to worker assignment problems for virtual cellular manufacturing systems. Rezazadeh et al. (2011) presented a mathematical model for the virtual cell formation problem by considering multi period planning horizon. The model, which considers real world instances, cannot be solved optimally within a reasonable amount of computational time. Therefore, they proposed a linear programming embedded particle swarm optimisation algorithm with a simulated annealing-based local search engine to solve the model.

On the other hand, some studies considered dynamic part demand changes, but under an assumption related to the main phases of cellular manufacturing. They assumed that the family types of parts are known when the parts enter the manufacturing system. Kannan and Ghosh (1996), Kannan (1997), Kannan (1998), Vakharia et al. (1999), Suresh and Slomp (2005), Nomden and Zee (2008), and Kesen et al. (2009) presented studies using this assumption.

In actual fact, determining the family types by considering the parts at the beginning of the solution is not enough in dynamic systems. This is because the part spectrum and the part family type of a part can be changed according to the changing part spectrum over time.

As seen, there is no integrated study which considers both the main phases of virtual cellular manufacturing and dynamic part demand arrivals effectively. Some studies which consider dynamic part demand arrivals work under the assumption of having knowledge of the family types of parts at the beginning of the solution. Some create virtual cells for parts instead of part families. However, removing this assumption and creating virtual cells for part families, and considering dynamism in part demand arrivals are important issues. This is because one of the most important aims of virtual cellular manufacturing is to provide efficiency in volatile manufacturing environments, and the other is to take advantage of grouping similar parts, such as reduced setup times and reduced lead times. In this thesis study, we removed this assumption and determined the virtual cells by considering part families via the presented agent based algorithm. The proposed approach enables us to realise part family formation, virtual cell formation, and scheduling simultaneously while considering dynamic part demand arrivals.

CHAPTER 3

AGENT BASED DYNAMIC PART FAMILY FORMATION IN CONCEPTUAL LEVEL

In this chapter a novel agent based clustering algorithm is presented for part family formation in cellular manufacturing by considering dynamic demand changes. However, it is not easy to directly compare the performance of the proposed algorithm with the literature results as there is no benchmark for dynamic cell formation problems. We attempt to compare the performance of the present algorithm on static test problems by dynamically introducing parts in these datasets to our algorithm.

As mentioned in the literature review, there is only one study which considers dynamic part arrivals in the part family formation problem in cellular manufacturing without the assumption of known part types at the beginning of the problem solution. This study was presented by Ben-Arieh and Sreenivasan (1999). Although Ben-Arieh and Sreenivan's (1999) methodology has several advantages, it can be improved in terms of handling dynamism more efficiently. Their algorithm consists of two separate phases as initial part family formation and negotiation. Negotiation starts among the agents after the initial part family formation is finished. Initial part family formation phase continues until the predetermined time is full. So, parts find the more appropriate part family for themselves after the initial part family formation phase. Time dependency is one of the features of most of the real world dynamic problems, and the solutions need to be found in response to the incoming information and track the optimal solutions through time as closely as possible (Psaraftis, 1995; Bianchi, 2000; Branke, 2001; Younes, 2006; Erol et al., 2012). In the thesis study, we considerably modified the algorithm proposed by Ben-Arieh and Sreenivasan (1999) in order to have a method to handle the dynamism more effectively. In our algorithm, the most important change is that the initial part family formation and negotiation phases are combined in order to obtain a more efficient dynamic approach, which tracks optimum or near optimum solutions closely. Consequently, any part can obtain the more appropriate part family for itself at any time considering existing conditions. Also by the proposed algorithm, besides the dynamic part demand arrivals, dynamic part demand cancellations can be handled efficiently. In addition, the proposed algorithm mainly considers the same calculations with Ben-Arieh and Sreenivan's (1999) methodology but it has a different auction based negotiation mechanism. Wellman (1993) suggested that in order to maximize the overall system performance, one of the efficient coordination mechanisms is market oriented programming, a concept which he initially introduced to the literature. In market oriented programming approach, negotiation among the agents are realized as the auctions in the real life. The activities and resource allocations for agents are derived by computing the competitive equilibrium of an artificial economy (Wellman, 1993).

In this chapter, the developed algorithm for dynamic part family formation problem is presented removing the assumption of having the knowledge of multi period plans.

3.1 Agent based dynamic part family formation

3.1.1 Problem definition

One of the fundamental problems in cellular manufacturing is part family formation. Part families are formed according to processing requirements of parts. In the literature, machining operations of the parts are considered as the processing requirements at conceptual level in cell formation (Gonçalves and Resende, 2004). This has been represented by a 0-1 machine-part incidence matrix. In 0-1 machine-part incidence matrix, m rows indicate m machines and p columns illustrate p parts. Each 0-1 instance in machine-part incidence matrix [A] as illustrated in Figure 3.1 determine a relationship between machines and parts: $a_{1,2}$=1 indicates the visit of part 2 to machine 1, and $a_{1,3}$=0 indicates that part 3 does not visit machine 1 etc.

		Parts								
		1	2	3	4	5	6	7	8	9
Machines	1	1	1	0	0	0	1	1	0	1
	2	0	1	1	0	1	1	1	0	1
	3	0	1	1	0	1	0	1	0	0
	4	0	0	1	1	0	0	0	1	0

Figure 3.1 Machine-part incidence matrix

For small size problems, part families can be detected by visual inspection, rearranging the rows and columns appropriately, using the machine-part incidence matrix, but visual inspection is not sufficient for larger size problems (Wang and Rose, 1997). One of the basic mathematical models for static part family formation was proposed by Kusiak et al. (1986) assuming the number of part families and size of each part family are known, is as follows (Sultan and Fedjkı, 1997):

$$min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{p} d_{ij} x_{il} x_{jl}$$

$$\sum_{j=1}^{p} x_{ij} = 1 \qquad i=1,...,n$$

$$\sum_{i=1}^{n} x_{ij} = m_j \qquad j=1,...,p$$

$$x_{ij} \in \{0,1\} \qquad i=1,...,n \qquad j=1,...,p$$

where x_{ij}=1 if part i belongs to part family j, and x_{ij}=0 otherwise. n and p indicate the number of parts and part families respectively. m_j represents the size of part family j. d_{ij} denotes the distance between part i and j.

It is not possible to handle dynamic part arrivals with classical mathematical programming approaches which were proposed for static clustering. Furthermore, other methods developed for static clustering are not sufficient. For example, one of the widely used successful clustering algorithms is k-means clustering algorithm proposed by MacQueen (1967). The algorithm mainly considers a set of individuals for clustering and processes in order to obtain clusters. However, when there is any change in the set of individuals, then the algorithm should start to process from the beginning. And also one needs to determine the number of clusters before the algorithm starts to work. If the problem considered has larger size and the environment is highly dynamic, then the algorithm may not be sufficient for the solution. But in the proposed algorithm dynamic arrivals can be handled without having to start perform from the beginning. And there is no need to determine the number of clusters, since it is determined during the execution of the algorithm dynamically.

As mentioned before, one of the important aims of this study is to model dynamic part arrivals. So it is assumed that all the parts in the given machine-part incidence matrix enter the system dynamically. And as they come into the system they are clustered by the agent based dynamic part family formation algorithm which is explained below.

3.2 Agent based dynamic part family formation algorithm

In the proposed agent based dynamic part family formation algorithm parts and part families are defined as agents. And also a manager agent is defined in order to manage the part family formation process. During the negotiation process of Ben Arieh and Sreenivasan's (1999) algorithm, part families bid out their parts respectively according to the average distances. The part family which has a maximum average distance bids out its part firstly. And the family bidding out its part bids out its farthest part. However, in the proposed algorithm there are no priorities as in the above. The opportunity to change their part families is given to all the parts satisfying the conditions explained in below. The part which gets the maximum bid wins the auction. Negotiation between part family agents is acted if any part agent detects there is a more appropriate part family for itself. The flowchart of the developed algorithm is given in Figure 3.2. The details of the algorithm such as calculations, updates, and the auction mechanism are presented in the following.

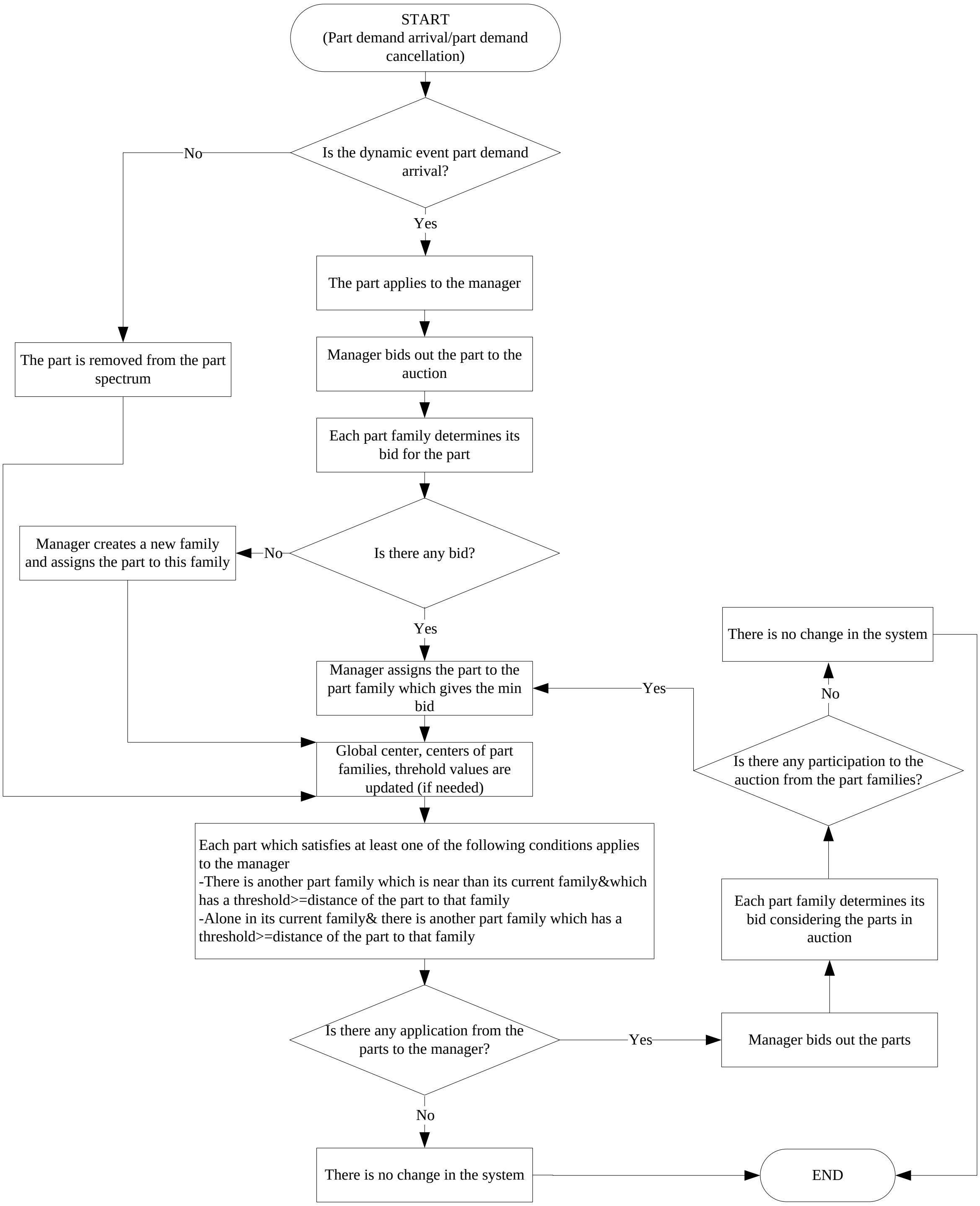

Figure 3.2 Flowchart of the agent based dynamic part family formation algorithm

In the proposed agent based dynamic part family formation algorithm, Equations 1-5 which were used by Ben Arieh and Sreenivasan (1999) are considered. These equations (Ben Arieh and Sreenivasan, 1999) have been re-written considering a number of attributes and Manhattan distance measure in this thesis study. Ben Arieh and Sreenivasan (1999) used Euclidean distances in these equations. In this study, Manhattan distances have been considered. In the proposed algorithm, threshold values of part families are calculated dynamically using the distance between the center of the part family and global center of the system. One of the conditions for acceptance to a part family is based on this threshold value. Since Manhattan distance is always greater than or equal to the Euclidean distance considering the same points, the calculated threshold value using Manhattan distance is always greater than or equal to the other one. It is clear that acceptance condition of a part to a family can be changed depending on the employed distance measure. So the appropriate distance measure should be used for the problem at hand. In the present study, we gathered results by using Euclidean distance measure for the test problems, and we observed that using Euclidean distance measure increased the number of part families in most of the test problems. Moreover, this situation also increased the possibility of having singleton part families (part family having less than two parts), although in the proposed algorithm there is an encouragement to destroy part families having one part. Thus, Manhattan distance measure is preferred.

Distance between part i and part j is calculated by Equation 3.1. $x_{i,a}$ illustrates the value of attribute a of part i ($a \in A$, (A: set of attributes)). In this study, machine requirements of parts are considered as the attributes of parts. For example, if part 1 visits machine 2, then attribute 2 value of part 1 ($x_{1,2}$) is equal to 1, otherwise the attribute 2 value of part 1 is equal to 0. Distance between part p and part family k with the center of $C_{k,a}$ are calculated using Equation 3.2.

$$d_{ij} = \sum_{a=1}^{A} \left| x_{i,a} - x_{j,a} \right| \tag{3.1}$$

$$D_{pk} = \sum_{a=1}^{A} \left| x_{p,a} - C_{k,a} \right| \tag{3.2}$$

Center of part family k (having n_k parts) and global center of the system (having N parts) for attribute a are determined using Equations 3.3 and Equation 3.4, respectively.

$$C_{k,a} = \frac{1}{n_k} \sum_{i=1}^{n_k} x_a \tag{3.3}$$

$$G_a = \frac{1}{N}\sum_{i=1}^{N} x_a \tag{3.4}$$

Threshold of part family k is calculated using Equation 3.5.

$$Threshold_k = \sum_{a=1}^{A} \left| G_a - C_{k,a} \right| \tag{3.5}$$

3.3 Agent based dynamic part family formation simulation model

The model which is created in AnyLogic^{R}, especially to illustrate the auction mechanism between agents is presented. Statecharts of part, manager and part family agent are given in Figure 3.3 which are created in AnyLogic^{R}. The statecharts in Figure 3.3 represent the possible states and transitions.

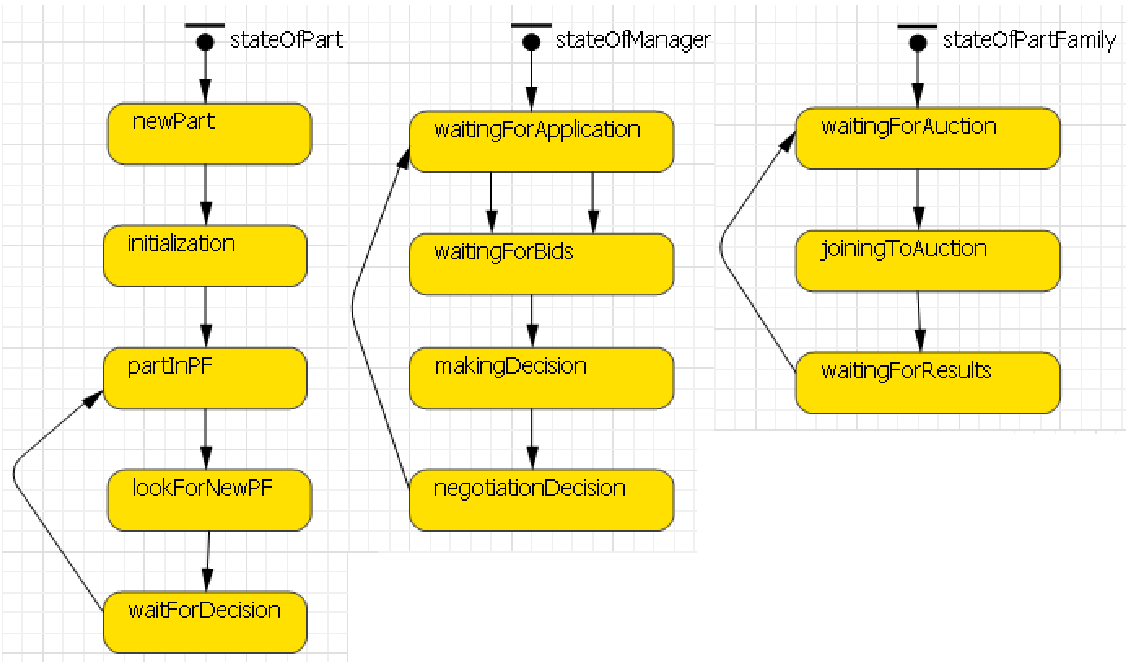

Figure 3.3 Statecharts of part, manager, and part family agent respectively.

When a part enters to the system dynamically it is placed in *newPart* state. If there is no process in the system it moves to *initialization* state and sends a message to the manager to inform its arrival. When manager receives the message, it sends a message to each part family to call them to the auction and manager moves from *waitingForApplication* state to *waitingForBids* state. The part family which receives a message from manager moves from *waitingForAuction* state to *joiningToAuction*, and determines its decision considering the part/parts in auction. It then moves from *joiningToAuction* state to *waitingForResults* state. After all the part families place to *waitingForDecision* state, the manager moves from *waitingForBids* state to *makingDecision* state. When manager makes its decision, assignments and updates are performed according to the decision made. The part moves from initialization state to *partInPF* state (if it is a new part). Manager moves from *makingDecision* state to *negotiationDecision* state. If any changes occur in the system, manager decides to communicate to each part to find out whether there is a more appropriate part family for the part than the current one. Then parts, part families, and clustering manager move to *partInPF* (if it is not a new part), *waitingForAuction*, and *waitinForApplication* states, respectively. If there is a decision for parts to find a new part family, each part moves from *partInPF* state to *lookForNewPF* state. If a part finds a more appropriate part family, it applies to the manager. After all the parts place to *waitForDecision* state, manager agent sends a message to each part family to call them to the auction. And the process continues as mentioned above.

When a demand cancellation occurs, the cancelled part is removed from the system (if it is alone in its family, the family is also destroyed) and the global center and threshold values are updated. And then if there is any request from parts for auction due to this change, the manager starts the auction process.

Auction between part families and manager is performed as follows:

```
Manager sends a message to the part families for calling them to the
auction
for each part family
   for each part in the auction
    if(part p has no family)|| (part p in the auction is alone in its
       part family k)
       Part family m calculates its bid for part p if condition 1 and
       condition 2 are satisfied
       Condition 1: Resulting configuration has not been reached before
                    in this   auction
       Condition 2: Distance between part p and part family m
       (Dpm)<=Thresholdm
    else
       Part family m calculates its bid for part p if condition 1,
       condition 2, and condition 3 are satisfied
       Condition 1: Resulting configuration has not been reached before
                    in this auction
       Condition 2: Dpm<=Thresholdm
       Condition 3: Dpk>Dpm
     endif
   endfor
   Part family m selects part p which has the min distance to itself
   (minimum calculated bid) among the parts in auction. And join to the
   auction for it
endfor
if there is any participation to the auction
   Manager determines the winner part p which gets the min bid among the
   parts in auction and the winner part family m which gives the min bid
   to it
   Manager assigns the winner part p to the winner part family m
   if number of parts in part family k equals to zero
     Manager destroys family k
     Family m updates its center and threshold
   else
     if part p in auction has no family
       Manager updates global center
       Each part family updates its threshold
       Family m updates its center
     else
       Family k and family m update their centers and thresholds
     endif
   endif
else
   if part p in auction has no family
     Manager creates a new part family n and assigns the part p to it
     Manager updates global center
     Each part family updates its threshold
     Family n updates its center
endif
```

3.4 An illustrative example

The proposed dynamic part family formation algorithm is presented on a simple problem which is created synthetically in order to explain the working of the algorithm. In the example there are ten machines. Parts enter to the system dynamically, and as they enter to the system they try to find the appropriate part family to themselves by utilizing the dynamic part family formation algorithm. Let's look at the system at some times in order to see the steps of the proposed algorithm in more detail.

The situation after the arrival of seven parts to the system dynamically is as follows: Machines (machine set M={1,2,…,10}) and parts (part set P=(1,2,…,7)) in the current system define the machine-part incidence matrix as shown in (Figure 4). Machine requirements of parts are considered as the attributes of parts in such a way that if $a_{1,2}$=1, then first attribute value of part 2 is equal to 1, otherwise (if $a_{1,2}$=0) the first attribute value of part 2 is equal to 0. Parts are numbered according to their attending order to the system in Figure 3.4. Dynamically generated part families are shown in Figure 3.5. In Figure 3.5, centers of the part families are given in terms of 10 dimensional arrays (each dimension indicates the calculated center value by considering each attribute (machine) with respect to M={1,2,…,10}).

Machines \ Parts	1	2	3	4	5	6	7
1	1	1	0	0	0	0	1
2	0	1	0	0	1	0	1
3	0	1	0	0	1	0	1
4	0	0	0	0	0	0	0
5	1	0	1	0	0	1	1
6	1	0	1	0	0	1	0
7	0	0	1	0	0	1	0
8	0	0	0	1	0	1	0
9	1	0	0	1	1	1	0
10	0	0	0	1	0	0	0

Figure 3.4 Machine-part incidence matrix.

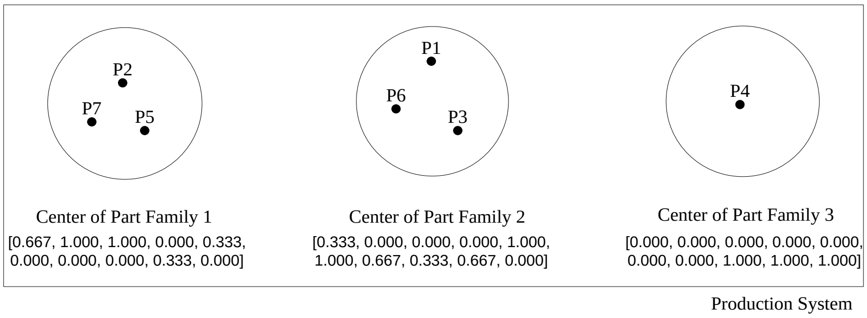

Figure 3.5 Dynamically generated part families.

Part 8 enters to the current system. The value of part incidence matrix for part 8 is $a_{i,8}{}^T = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$. Part 8 sends a message to the manager to inform manager of its arrival. The manager sends a message to each part family to call them to the auction. Threshold value of the part families and distance of part 8 from part families is shown in Figure 3.6.

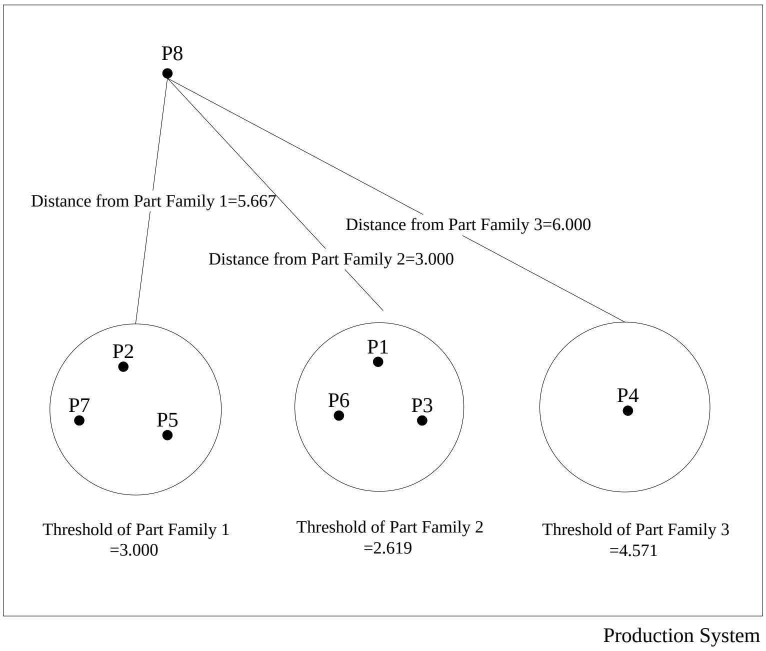

Figure 3.6 Threshold value of part families and distance of part 8 from the part families.

Since part 8 is acceptable for none of the part family, there is no bid for part 8. The manager created part family 4 and part 8 is assigned to part family 4 as illustrated in Figure 3.7.

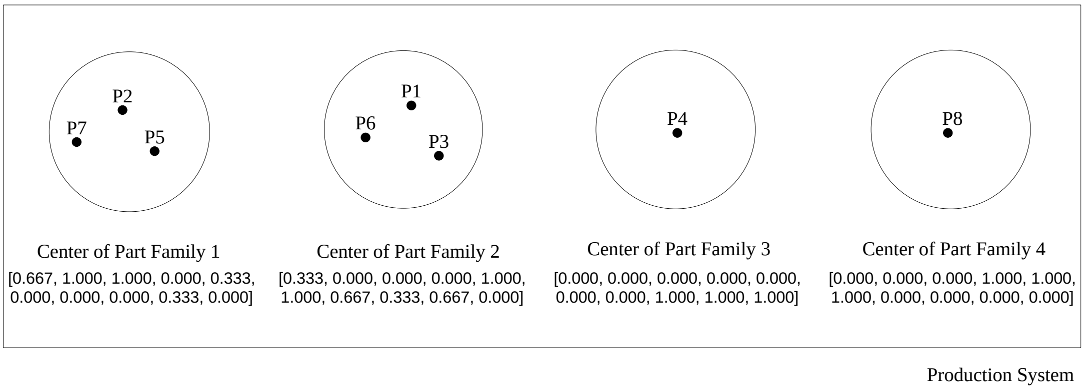

Figure 3.7 Dynamically created part families

All parts check if there is a more appropriate part family themselves than the current one. Since there is no part family for any part satisfying conditions, no change occurs in the families of parts.

Part 9 enters to the system. The value of part incidence matrix of part 9 is $a_{i,9}{}^{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. Part 9 sends a message to the manager to inform manager of its arrival. The manager sends a message to each part family for calling them to the auction. Threshold value of the part families and distance of part 9 from part families is shown in Figure 3.8.

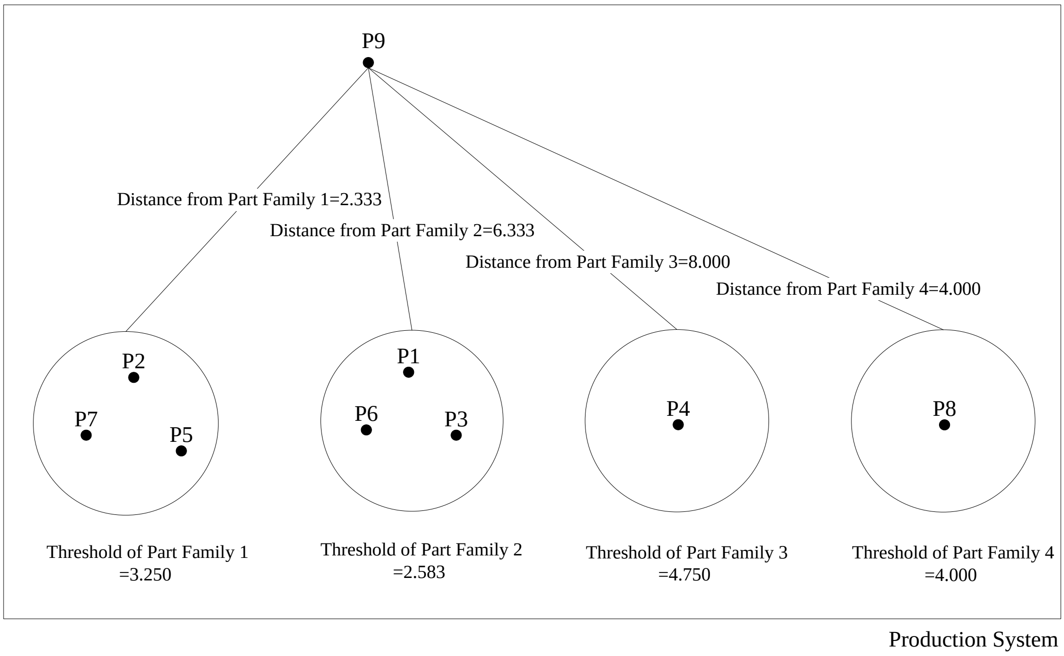

Figure 3.8 Threshold value of part families and distance of part 9 from the part families.

Part family 1 determines a bid for part 9 as 2.333. Since there is only one bid, the manager assigns part 9 to the part family 1. The new configuration is shown in Figure 3.9.

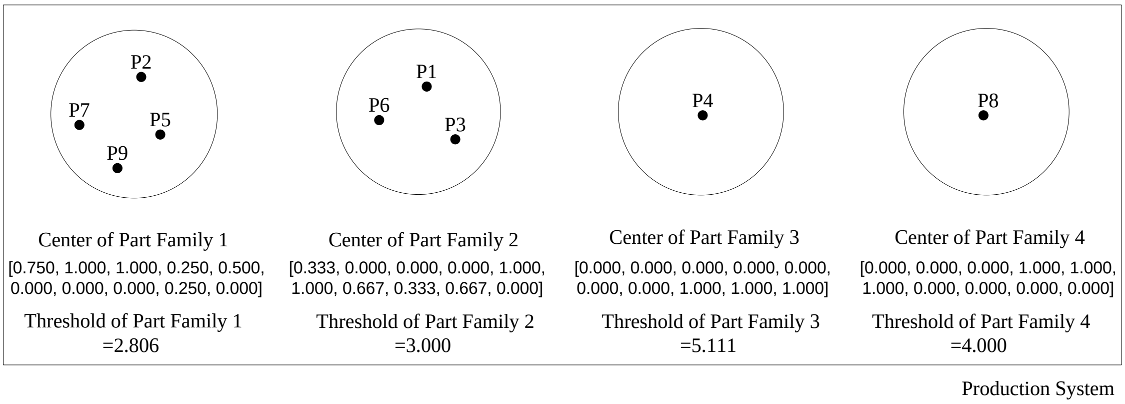

Figure 3.9 Dynamically created part families.

All parts check if there is a more appropriate part family themselves than the current one. Part 8 applies to the manager. Manager bids out part 8. Part family 2 determines a bid for part 8. Manager assigns part 8 to the part family 2. Part family 4 has no part any more, therefore manager destroys it. The new configuration of the part families is shown in Figure 3.10.

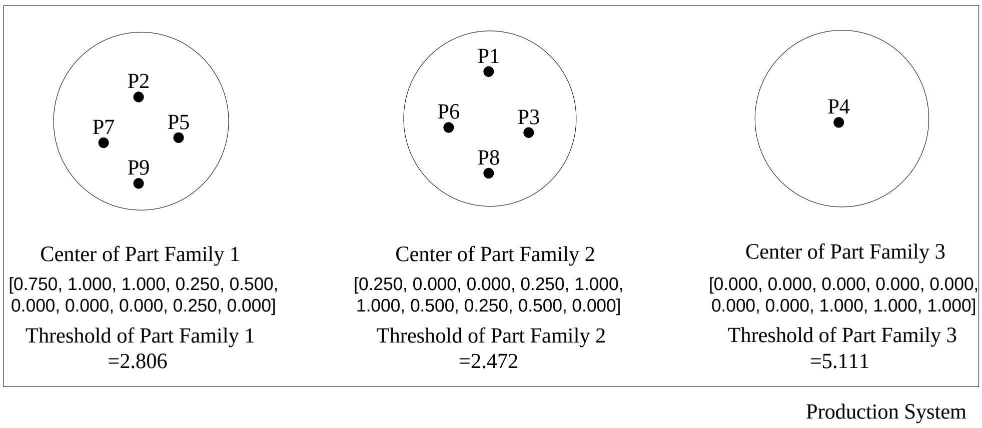

Figure 3.10 Dynamically created part families.

All parts check if there is a more appropriate part family themselves than the current one. Since there is no part family for any part satisfying conditions, no change occurs in the families of parts.

Parts continue to enter: at simulation time 340.00, there are 18 parts in the system. Machines (machine set M={1,2,…,10}) and parts (part set P=(1,2,…,18)) in the current system are shown through the machine-part incidence matrix in Figure 3.11. Figure 3.12 shows the final configuration of the system.

Machines \ Parts	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	0	0	0	0	1	0	1	1	0	0	1	1	0	1	0	0
2	0	1	0	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0
3	0	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0	0
4	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0
5	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0
6	1	0	1	0	0	1	0	1	0	0	1	1	0	0	1	0	1	0
7	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
8	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	0	1
9	1	0	0	1	1	1	0	0	0	0	1	1	0	0	0	1	0	1
10	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1	0	1

Figure 3.11 Machine-part incidence matrix.

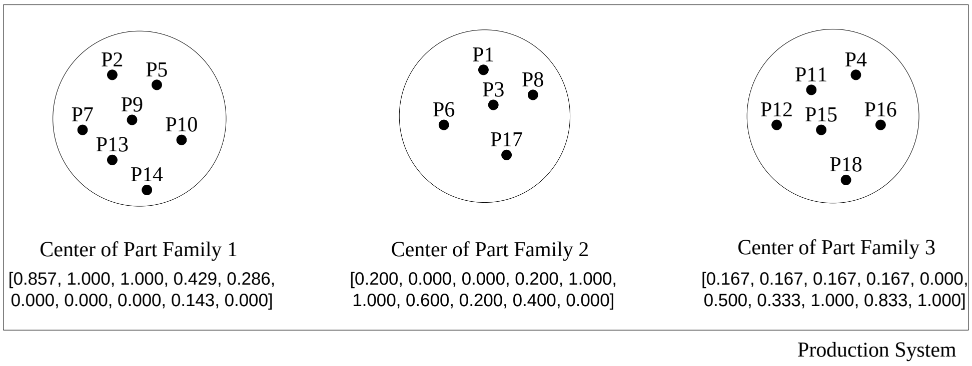

Figure 3.12 Dynamically created part families.

3.4 Computational study

Since there is no comparable result for dynamic part family formation problem in the literature we have tried to compare the present agent based algorithm's results with the results of the test problems generated using traditional cell formation (static) approaches. In order to enable our algorithm to work with static test problems we have dynamically introduced parts in these data sets to our algorithm allowing random attending orders. By this comparison, we aimed to investigate whether our algorithm can find efficient solutions to the problems while tracking and evaluating the changes. In order to be able to compare the results, machines need to be allocated to the dynamically created part families. Allocation of machines to the part families is performed using the procedure proposed by the Wu et al. (2008) which is given in the following. And an example is given in Appendix A.

```
Consider the results of the agent based dynamic part family
formation algorithm.
Repeat (until all machines assigned)
    Determine the cell to which the machine allocation will result
    in the least sum of number of voids and exceptional elements. If
    a tie occurs, assign the machine to a cell with the least number
    of voids.
```

In the procedure exceptional elements are 1's outside the diagonal blocks, and 0's inside the diagonal blocks are called voids.

Several measures of efficiency of cell formation have been proposed in the literature. However, grouping efficacy proposed by Kumar and Chandrasekharan (1990) comes into prominence with several reasons; it is a widely accepted measure in the literature to evaluate the goodness of the proposed algorithms; it considers both the within-cell utilization and inter-cell movement; it can discriminate well-structured and ill-structured matrices, etc. (Gonçalves and Resende, 2004). Therefore, grouping efficacy is employed in this study in order to determine the efficiency of the proposed algorithm. Grouping efficacy can be computed by making use of Equation 3.6 (Kumar and Chandrasekharan, 1990):

$$Grouping\ Efficacy = \frac{e - e_0}{e + e_v} \tag{3.6}$$

Where e is the total number of 1's in the matrix, e_0 is the total number of exceptional elements and e_v is the total number of voids.

The performance of the proposed algorithm is compared with several classical approaches, namely ZODIAC (Chandrasekharan and Rajagopalan, 1987), GRAFICS (Srinivasan and Narendran, 1991), MST (Srinivasan, 1994), TSPGA (Cheng et al., 1998), GA (Onwubolu and Mutingi, 2001), GPA (Dimopoulos and Mort, 2001), HGGA (James et al., 2007), AS (Islier, 2005), ACRS (Kao and Li, 2008), PACO (Megala et al., 2008), and ACO-CF (Xiangyong et al., 2010) (results of ACO-CF with constraint not allowing residual cells are represented as ACO-CF (1) in Table 3.2). (The results of the previously mentioned algorithms are taken from the study of Xiangyong et al. (2010)). Most of these algorithms are metaheuristics aiming to provide optimal solutions for these static problems. One important point during the comparison process analyzed by Xiangyong et al. (2010) is the effect of having residual cells (cells having only machines or parts) in the solution to the efficacy measure. In their study, they also gave place to the results obtained using their proposed algorithm allowing residual cells. They expressed that efficacy can increase while allowing the residual cells in diagonal blocks. The results obtained using ACO-CF (Xiangyong et al., 2010) allowing residual cells (represented as ACO-CF (2) in Table 3.2) are also given in Table 3.2. One of the constraints used in some of the studies (Gonçalves and Resende, 2004) is not allowing singletons (cells having less than two parts or two machines) and Gonçalves and Resende (2004) mentioned that this constraint also degrades the performance of the algorithm.

Our algorithm allows occurring singleton part families although there is an encouragement to destroy part families which has only one part. Since parts enter the system dynamically, one part family having only one part can be a part family having two or more parts after the entrance of a new part to the system. So in this open system, singleton part family constraint will be meaningless. A machine allocation procedure which does not affect the obtained part families is required, since the main goal of this study is dynamic part family formation. To this aim, a machine allocation procedure which is independent of part family formation phase was used. The procedure used for allocation of machines to the part families allows residual cells. In the comparisons we also give place to the results which were obtained using the simulated annealing based approach (SACF) of Wu et al. (2008) from where we adopted machine allocation procedure. Data sets for the test problems are obtained from Gonçalves and Resende (2004). We also compare our results with theirs.

Properties of test problems are given in Table 3.1 and a direct comparison of the results is given in Table 3.2. There are 35 test problems. Bolded values in Table 3.2 represent the results which are smaller than or equal to results generated by agent based dynamic part family formation algorithm. Apart from three problems (problems: 9, 28, and 34) the agent based dynamic approach is able to produce the same or better results than the static algorithms available in the literature. Here we should again mention that the proposed agent based algorithm is not an optimization based algorithms (i.e. it does not conduct a search procedure for a given problem as it adaptively constructs a solution as parts enter to the system). Therefore, the results are exciting and show that the proposed algorithm has an ability to follow optimal solutions in dynamic circumstances. Moreover, as can be seen from Table 3.2, the proposed algorithm dominates Islier's (2005) ant system algorithm for the compared problems. A similar situation exists for Onwubolu and Mutingi's (2001) genetic algorithm approach, agent based approaches produced better or same results for 19 test problems within 25 test problems. These results are especially important as ant colony and genetic algorithms are known to be members of the most powerful optimization algorithms in the literature.

Table 3.1 Properties of the test problems

No.	Source of test problem	Size
1	King and Nakornchai (1982)	5x7
2	Waghodekar and Sahu (1984)	5x7
3	Seifoddini (1989)	5x18
4	Kusiak and Cho (1992)	6x8
5	Kusiak and Chow (1987)	7x11
6	Boctor (1991)	7x11
7	Seifoddini and Wolfe (1986)	8x12
8	Chandrasekharan and Rajagopalan (1986a)	8x20
9	Chandrasekharan and Rajagopalan (1986b)	8x20
10	Mosier and Taube (1985a)	10x10
11	Chan and Milner (1982)	10x15
12	Askin and Subramanian (1987)	14x24
13	Stanfel (1985)	14x24
14	McCormick et al. (1972)	16x24
15	Srinivasan, Narendran, and Mahadevan (1990)	16x30
16	King (1980)	16x43
17	Carrie (1973)	18x24
18	Mosier and Taube (1985b)	20x20
19	Kumar, Kusiak, and Vannelli. (1986)	20x23
20	Carrie (1973)	20x35
21	Boe and Cheng (1991)	20x35
22	Chandrasekharan and Rajagopalan (1989)	24x40
23	Chandrasekaran and Rajagopalan (1989)	24x40
24	Chandrasekharan and Rajagopalan (1989)	24x40
25	Chandrasekharan and Rajagopalan (1989)	24x40
26	Chandrasekharan and Rajagopalan (1989)	24x40
27	Chandrasekharan and Rajagopalan (1989)	24x40
28	McCormick, Schweitzer, and White (1972)	27x27
29	Carrie (1973)	28x46
30	Kumar and Vannelli (1987)	30x41
31	Stanfel (1985)	30x50
32	Stanfel (1985)	30x50
33*	King and Nakornchai (1982)	36x90
34	McCormick et al. (1972)	37x53
35	Chandrasekharan and Rajagopalan (1989)	40x100

*Although the size of problem is mentioned as 36*90, there are 30 machines in the data set

Table 3.2 Comparison of results

	Grouping efficacy (%)														
No.	ZODIAC	GRAFICS	MST	TSPGA	GPA	GA	EA	HGGA	AS	ACRS	PACO	ACO-CF (1)	ACO-CF (2)	SACF	The proposed approach
1	**73.68**	**73.68**					**73.68**	82.35	**73.68**	82.4	**73.68**	82.35	82.35		75.00
2	**56.52**	60.87		68.00		62.50	62.50	69.57		68.0	69.57	69.57	69.57	69.57	60.00
3				**77.36**		**77.36**	**79.59**	**79.59**			**79.59**	**79.59**	80.85	**79.59**	79.59
4				**76.92**		**76.92**	**76.92**	**76.92**			**76.92**	**76.92**	79.17	**76.92**	76.92
5	**39.13**	**53.12**		**46.88**		**50.00**	**53.13**	60.87			58.62	60.87	60.87	60.87	56.52
6				**70.37**		**70.37**	**70.37**	**70.83**			**70.37**	**70.83**	**70.83**	**70.83**	70.83
7	**68.30**	**68.30**					**68.30**	**69.44**			**68.29**	**69.44**	**69.44**		69.44
8	**85.24**	**85.24**	**85.24**	**85.24**	**85.20**	**85.24**	**85.25**	**85.25**		85.3	**85.25**	**85.25**	**85.25**	**85.25**	85.25
9	58.33	58.13	58.72	58.33	58.70	55.91	58.72	58.72			58.72	58.72	58.72	58.41	36.96
10	**70.59**	**70.59**	**70.59**	**70.59**		72.79	**70.59**	75.00			**70.59**	75.00	75.00	75.00	70.59
11	**92.00**	**92.00**		**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**81.82**	**92.0**	**92.00**	**92.00**	**92.00**	**92.00**	92.00
12	**64.36**	**64.36**	**64.36**				69.86	72.06			69.86	72.06	73.13		65.75
13	**65.55**	**65.55**		**67.44**	71.80	**63.48**	**69.33**	71.83		67.1	70.51	71.83	72.86	71.21	69.33
14	**32.09**	**45.52**	**48.70**				52.58	52.75			51.96	52.75	53.26		48.91
15	**67.83**	**67.83**	**67.83**				**67.83**	68.99			**67.83**	68.99	69.92		67.91
16	**53.76**	54.39	54.44	**53.89**		86.25	54.86	57.53	**39.25**	**48.8**	54.86	57.53	58.04	**52.44**	54.27
17	**41.84**	**48.91**	**44.20**				54.46	57.73			54.96	57.73	57.73		51.00
18	**21.63**	38.26		**37.12**		**34.16**	42.96	43.18			42.75	43.45	43.97	41.02	37.93
19	**38.66**	49.36	**43.01**	46.62	49.00	**39.02**	49.65	50.81			49.65	50.81	50.81	50.81	43.48
20	**75.14**	**75.14**	**75.14**	**75.28**	76.70	**66.30**	76.22	77.91			78.40	77.91	78.88	78.40	76.14
21				**55.14**	56.80	**44.44**	58.07	57.98			58.38	57.98	58.60	**56.04**	56.35
22	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**70.47**		**100.00**	**100.00**	**100.00**	**100.00**	100.00
23	**85.10**	**85.10**	**85.11**	**85.11**	**85.10**	**85.11**	**85.11**	**85.11**	**61.49**		**85.11**	**85.11**	**85.11**	**85.11**	85.11
24	**37.85**	**73.51**	**73.51**	**73.03**	**73.50**	**73.03**	**73.51**	**73.51**	**49.71**		**73.51**	**73.51**	**73.51**	**73.51**	73.51
25	**20.42**	**43.27**	**51.81**	**49.37**	53.3	**37.62**	**51.97**	53.29	**35.75**		52.83	53.29	53.29	52.44	52.03
26	**18.23**	44.51	44.72	44.67	47.90	**34.76**	47.06	48.95	**32.08**		47.21	48.95	48.95	47.13	44.37
27	**17.61**	41.67	44.17	42.50	43.70	**34.06**	44.87	47.26	**31.00**		44.71	47.26	47.26	44.64	41.06
28	52.14	47.37	51.00				54.27	54.02			54.27	54.82	54.82		46.96
29	**33.01**	**32.86**	40.00				44.62	46.91			45.67	47.08	47.72		38.71
30	**33.46**	55.43	55.29	53.80	60.70	**40.96**	58.48	63.31			60.38	63.31	63.31	62.42	51.35
31	**46.06**	56.32	58.70	56.61	59.40	48.28	59.66	59.77			59.55	59.77	59.77	60.12	47.52
32	**21.11**	47.96	**46.30**	45.93	50.00	**37.55**	50.51	50.83			50.51	50.83	50.83	50.51	45.36
33	**32.73**	39.41	40.05				42.64	46.35			44.75	47.11	47.53		37.58
34	52.21	52.21					56.42	60.64			57.94	60.64	61.31		45.69
35	83.92	83.92	83.66	84.03	84.00	83.90	84.03	84.03	**39.56**	84.0	81.64	84.03	84.03	84.03	71.61

The success of the proposed algorithm in modeling the dynamic arrivals in part family formation is demonstrated by the above analysis. However, in real life manufacturers are also frequently faced with other types of dynamic events, such as demand cancellation and reentrance. The solution approach which models these dynamic events successfully is very precious. Since, in order to get ahead in this age of global competition, manufacturers need an approach which works under the operational dynamics. Our algorithm has abilities to manage these dynamic events. In order to examine these capabilities, a scenario is prepared. According to the scenario, differing from the above analyses, a dynamic event is not always a part demand. It can be a part demand or a cancellation with probabilities 0.80 and 0.20, respectively. If the dynamic event is determined as part demand according to the given probability, then the part in the given machine-part incidence matrix of the problem enters the system (allowing random attending orders) as realized in above analysis. But, if the dynamic event is determined as cancellation, then the cancelled part demand is determined dynamically among the parts in the system randomly. After all the parts in the problem are entered into the system once, the cancelled parts are reentered into the system. In Table 3.3 we presented the detailed results considering 10 independent runs of the proposed algorithm.

Table 3.3 Details of the results

No.	Results of proposed approach (Grouping efficacy (%))			Results of proposed approach with cancellation and reentrance (Grouping efficacy (%))		
	Minimum	Average	Maximum	Minimum	Average	Maximum
1	61.11	69.88	75.00	61.11	70.50	75.00
2	50.00	58.00	60.00	50.00	58.00	60.00
3	64.58	73.59	79.59	64.58	76.59	79.59
4	66.67	75.02	76.92	76.92	76.92	76.92
5	46.15	53.72	56.52	46.15	52.85	56.52
6	64.00	68.91	70.83	67.86	69.16	70.83
7	63.89	67.38	69.44	63.89	65.85	68.29
8	85.25	85.25	85.25	78.69	84.59	85.25
9	32.61	34.71	36.96	32.61	35.25	40.22
10	70.59	70.59	70.59	70.59	70.59	70.59
11	92.00	92.00	92.00	92.00	92.00	92.00
12	62.50	63.82	65.75	62.50	65.68	68.92
13	64.47	67.93	69.33	67.05	67.73	69.33
14	43.88	46.20	48.91	40.63	46.47	49.48
15	64.18	66.59	67.91	62.12	66.85	68.15
16	49.07	52.51	54.27	48.67	53.08	53.75
17	45.54	47.94	51.00	43.33	46.41	49.02
18	30.70	34.87	37.93	32.48	35.49	39.50
19	36.13	40.01	43.48	33.05	37.90	44.92
20	76.14	76.14	76.14	76.14	76.14	76.14
21	53.59	54.97	56.35	52.91	55.43	56.61
22	80.86	86.91	100.00	60.33	83.13	100.00
23	68.90	75.49	85.11	66.67	76.92	85.11
24	60.48	69.91	73.51	60.36	72.19	73.51
25	40.54	45.67	52.03	40.54	44.85	47.92
26	36.91	41.25	44.37	36.05	40.67	43.59
27	34.44	38.78	41.06	34.69	38.68	41.67
28	40.40	42.86	46.96	39.26	42.60	47.37
29	30.74	35.41	38.71	31.60	36.18	39.74
30	41.28	47.10	51.35	45.41	48.63	51.19
31	38.03	44.10	47.52	40.10	46.11	54.59
32	38.00	42.12	45.36	38.22	43.46	46.24
33	29.82	33.44	37.58	34.38	36.51	38.11
34	39.62	42.54	45.69	39.90	42.89	46.19
35	53.05	63.01	71.61	54.45	64.40	76.84

The results of the algorithm considering these two different dynamic cases are compared in order to prove the success of the algorithm on managing the cancellation and reentrance events. In the comparison paired-t test is used, since it is appropriate for comparing the average results of these two dynamic cases. A paired-t test matches the values of the two samples in a pairwise manner, computes the mean of differences between the pairs and then tests whether the mean of differences is different from zero. Shortly, the hypothesis of no difference between the two samples is tested by paired-t test. The assumption that must be satisfied in order to use paired-t test is that paired differences should follow a normal distribution. The assumption is satisfied in our comparison. In the comparison all the test problems are evaluated, so the sample size is 35. According to the test, p-value is determined as 0.155. Since the p-value is greater than 0.05, the hypothesis of no difference between samples is accepted. As seen the difference between the results of these two dynamic cases is not statistically significant. So we can say that the proposed algorithm can handle cancellation and reentrance situations successfully.

CHAPTER 4

AN OVERVİEW OF DYNAMIC AGENT BASED VIRTUAL CELLULAR MANUFACTURING

In this chapter, an overview of the proposed algorithm is presented. The proposed approach aims to handle the operational issues of manufacturing system and dynamism in part demand arrivals. The main manufacturing concepts that operate for the same purpose are gathered together and an integrated system is proposed. The main concepts and methods which are used for this purpose are illustrated in Figure 4.1.

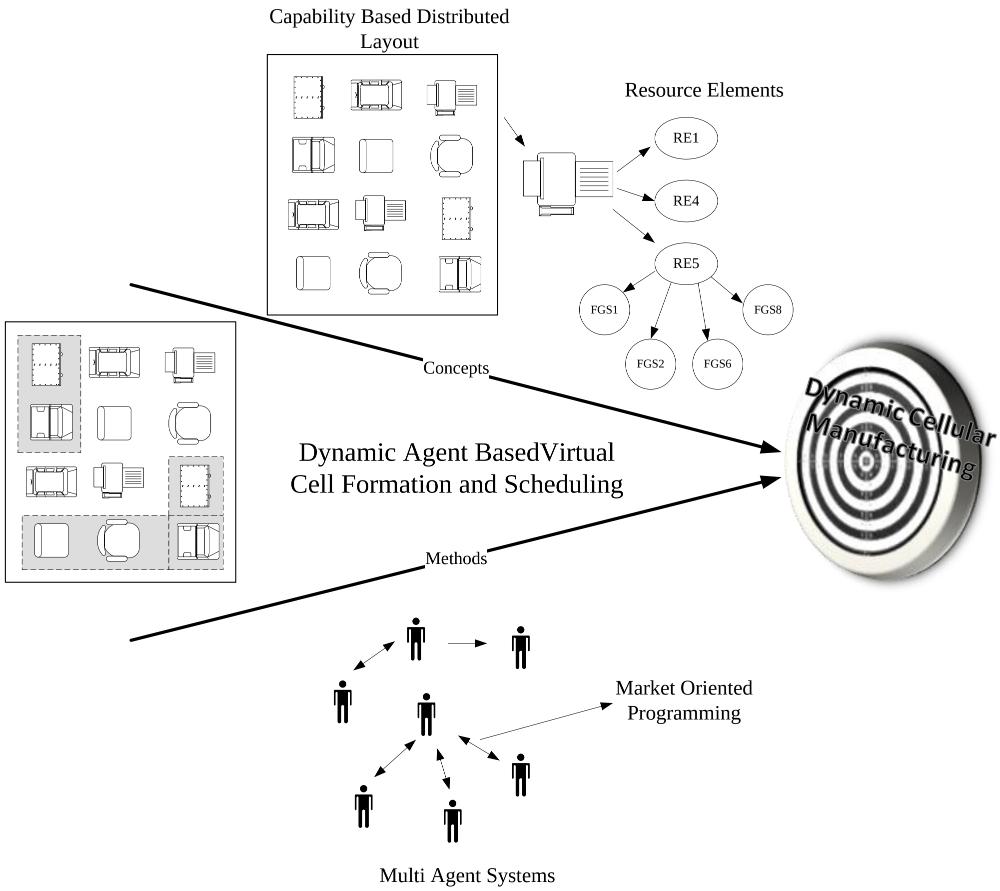

Figure 4.1 The main concepts and methods

Agent based modelling and virtual cellular manufacturing systems have been explained in the literature review section of the thesis study. Resource elements, capability based distributed layout and market oriented programming approaches are explained in the following.

4.1. Resource elements

In the manufacturing environment, parts can be assigned to machines if and only if the selected machines have the required capabilities of the parts. It is necessary to define the capabilities of machines and requirements of parts in a common way to make assignments properly. In this context, one of the methods used is the Resource Elements (REs) approach which was defined by Gindy et al. (1996). Gindy et al. (1996) reported that, according to their study, the use of resource elements provides better matching between the processing requirements of components and capabilities of machine tools compared with the conventional machine-based approach. In the resource elements approach, form generating schemas are used to define these capabilities. A form generating schema consists of a cutting tool, motion set, and technological output. The resource elements are determined by an iterative procedure which considers the form generating schemas.

Detailed information on form generating schemas and resource elements can be found in the studies of Gindy et al. (1996), Baykasoglu (1999), and Baykasoglu (2003). In our approach, the use of resource elements provides the opportunity to model flexibility in a better way.

4.2. Capability based distributed layout

For a successful manufacturing system under dynamic conditions, besides selected manufacturing strategy, determination of the most appropriate layout for the selected strategy is also an important issue. In highly volatile manufacturing environments, in which the part spectrum and demand change rapidly, a flexible layout is needed (Benjaafar and Sheikhzadeh, 2000; Baykasoglu, 2003). This is because, in a changing manufacturing environment, routings vary intensively and unplanned changes can occur, and the reconfiguration of layouts which are developed for a particular part spectrum is very difficult and expensive (Benjaafar and Sheikhzadeh, 2000; Baykasoglu, 2003).

Baykasoglu (2003) specified that the distributed layout approach is a better alternative for virtual cellular manufacturing applications. In the distributed layout approach, similar machines are scattered in the factory. In this way, the accessibility of the machines is increased from different regions of the layout. Therefore, the changing part spectrum can be handled with the distributed layout approach with acceptable material travel distances (Baykasoglu, 2003). For detailed explanations about the capability based distributed layout approach, refer to Baykasoglu (2003).

4.3. Market oriented programming

Although multi agent approaches provide lots of opportunities in many areas, control of agents is not an easy issue; it gets especially difficult depending on the size of the population of agents in the system (Flower, 2005). Moreover, if the system is an open system, controlling the system is even more difficult. This is because unknown agents enter to the system at unknown times (Flower, 2005).

In multi agent systems, agents act according to their own interests and benefits. Thus, in order to maximise the performance of the overall system, an efficient coordination mechanism between agents is required (Wellman, 1993). One of these coordination mechanisms is market oriented programming, which was first introduced by Wellman (1993) in the literature. In market oriented programming, solutions to distributed resource allocation problems are derived by computing the competitive equilibrium of an artificial economy (Wellman, 1993).

In this thesis study, the agent based dynamic virtual cell formation and scheduling algorithm is presented considering all these main concepts and approaches. The proposed algorithm consists of three main phases which progress simultaneously. These are the part family formation phase, virtual cell formation phase, and the scheduling phase. The transitions between these phases are also very important. Figure 4.2 shows the framework of the proposed approach.

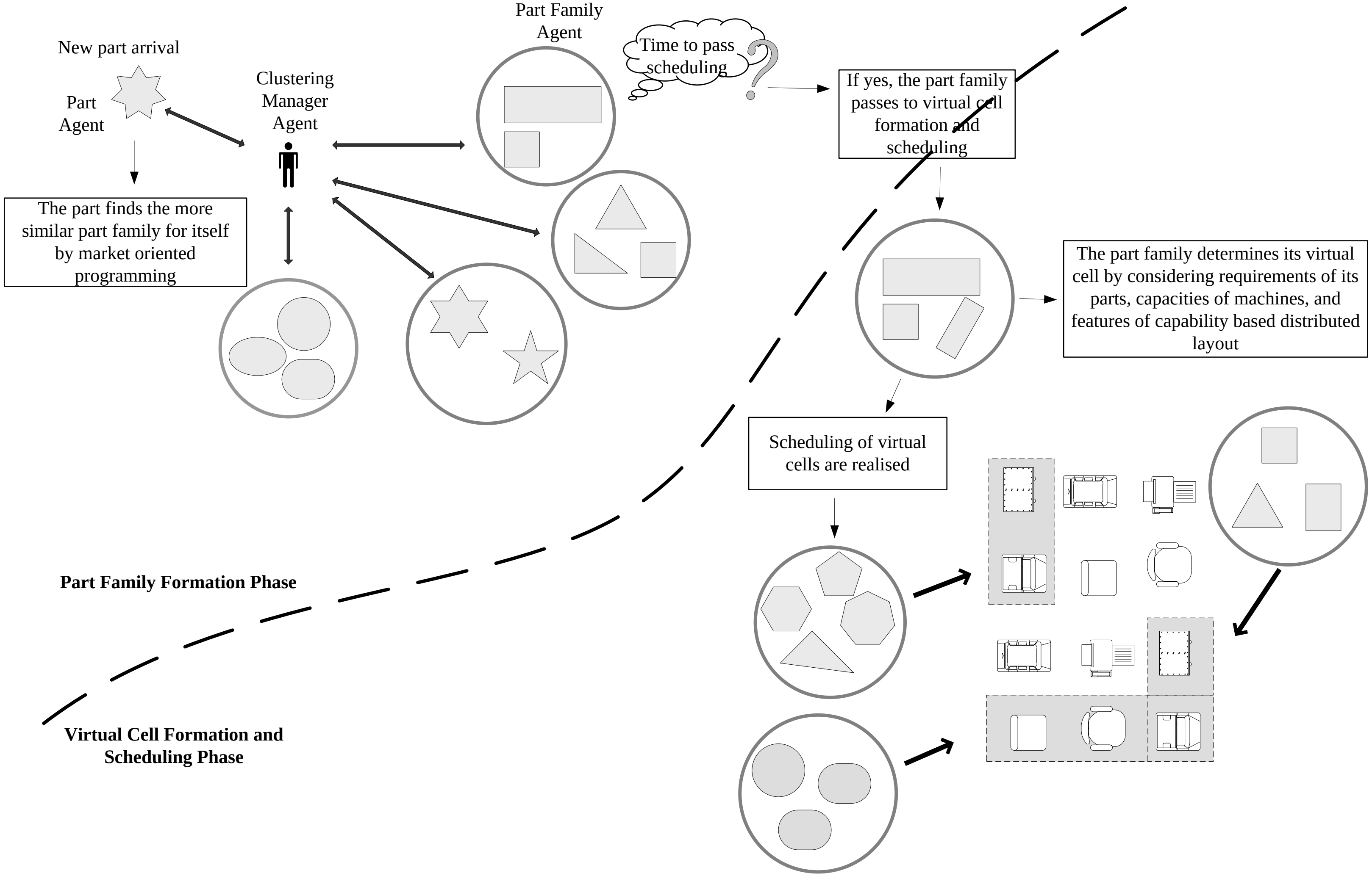

Figure 4.2 The framework of the proposed approach

The proposed approach is realised on AnyLogic^{R} platform which supports agent based modelling. As illustrated in Figure 4.2, in the model four types of agents are defined such as part, part family, clustering manager, and machine. We illustrate the main steps of the proposed algorithm by considering Figure 4.2 and the statecharts of the agents. A statechart which consists of states and transitions is a visual construct that enables us to define the event and time driven behaviour of agents (As of November 29, 2013, AnyLogic^{R} mentioned on its website http://www.anylogic.com/upload/Big%20Book%20of%20AnyLogic/Designing_state-based_behavior-statecharts.pdf). Statecharts created in AnyLogic^{R} for the parts, part families, clustering manager, and machines are given in Figures 4.3, 4.4, 4.5 and 4.6, respectively. As an example, some Java codes of the part agent is presented in Appendix B.

We examine the proposed approach by the arrival of a part demand to the system dynamically. The part agent is created by the arrival of the part demand and it is placed in the *newPart* state. If there is no process related to clustering issues in the system it moves to the *initialization* state. Some records and assignments are done in the *newPart* and *initialization* states, such as arrival time to the system and due date assignment. When the part is in the *initialization* state, it sends a message to the clustering manager to inform of its arrival.

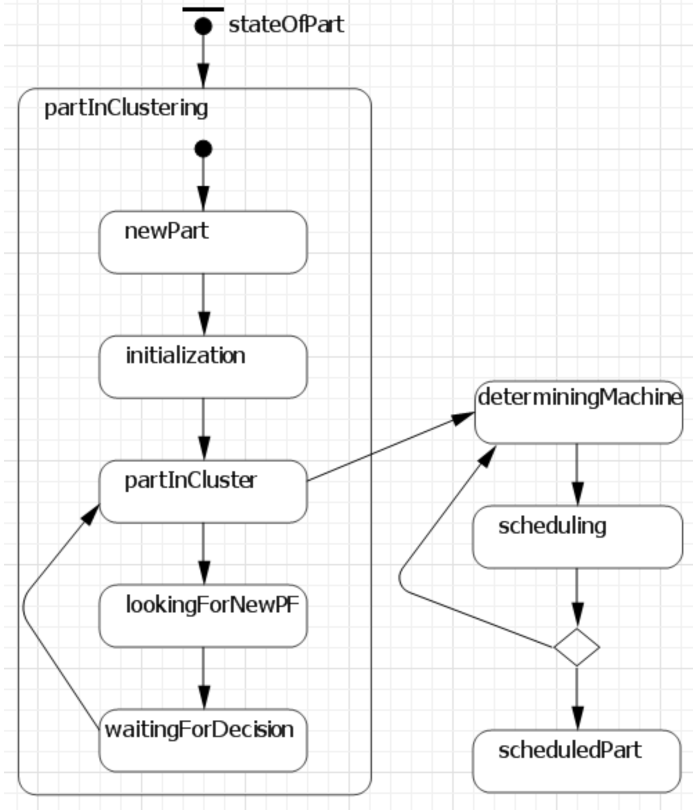

Figure 4.3 Statechart of a part agent

The clustering manager which receives the message from the part communicates with the part families. The manager sends a message to each part family to invite them to the auction. Then it moves from the *waitingForApplication* state to the *waitingForBids* state.

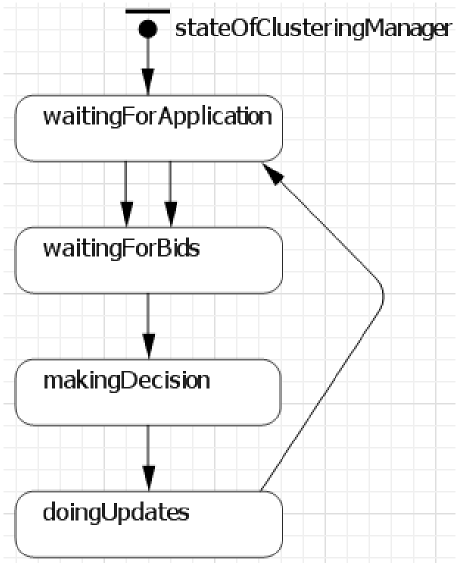

Figure 4.4 Statechart of the clustering manager agent

Each part family which receives a message from the clustering manager moves from the *waitingForAuction* state to the *joiningToAuction* state. In this state each part family makes a decision whether to join the auction in order to bid or not for the part in the auction. If it decides to join, then it determines its bid. It then moves to the *waitingForResults* state.

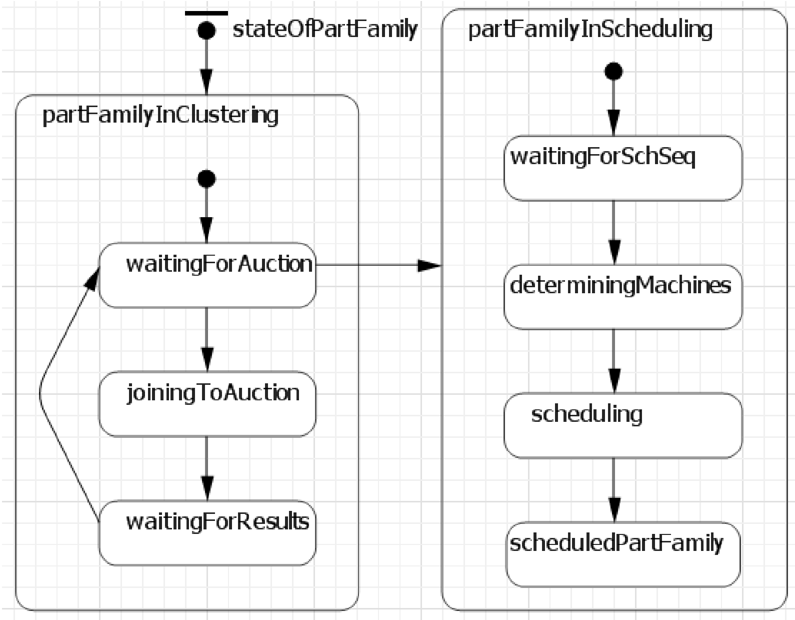

Figure 4.5 Statechart of a part family agent

When all the part families move to the *waitingForResults* state, then the clustering manager moves from the *waitingForBids* state to the *makingDecision* state. In this state, the clustering manager evaluates the bids. It then determines the part family for the part. After making the assignments, the manager moves to the *doingUpdates* state. The part and part families move to the *partInCluster* and *waitingforAuctionState*, respectively. If there is a change in the system, then the clustering manager calls each part to determine whether it wants to change its part family or not. Each part moves to the *lookingForNewPF* state in order to make the evaluation. Each part which desires to change its part family applies to the manager agent. If there is any application, the clustering manager restarts the auction process.

Part families continuously make controls to determine the right time for passing from the part family formation phase to the virtual cell formation and scheduling phase. A part family which decides to pass from the *partFamilyInScheduling* state according to this control moves to the *waitingForSchSeq* state, and then moves to the *determiningMachines* state when its turn arrives. In this state the part family determines the machines for the virtual cell by considering the requirements of its parts and the constraints of the manufacturing environment. After the determination it moves to the *scheduling* state. Parts of the part family leave the *partInCluster* state with the movement of the part family. The parts and the determined machines communicate for scheduling by considering the scheduling rules.

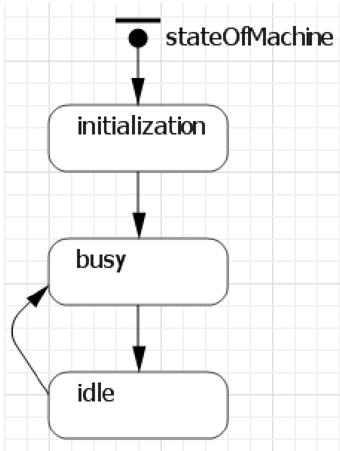

Figure 4.6 Statechart of a machine agent

After all the resource elements of a part are scheduled, the part moves to the *scheduledPart* state; after all the parts of a part family are scheduled, the part family moves to the *scheduledPartFamily* state. In this way, the part family formation, virtual cell formation and scheduling processes continue simultaneously as long as there are parts in the system. The interaction diagram of agents are illustrated in Figure 4.7, and a detailed explanation of the methodology is presented in Chapter 5.

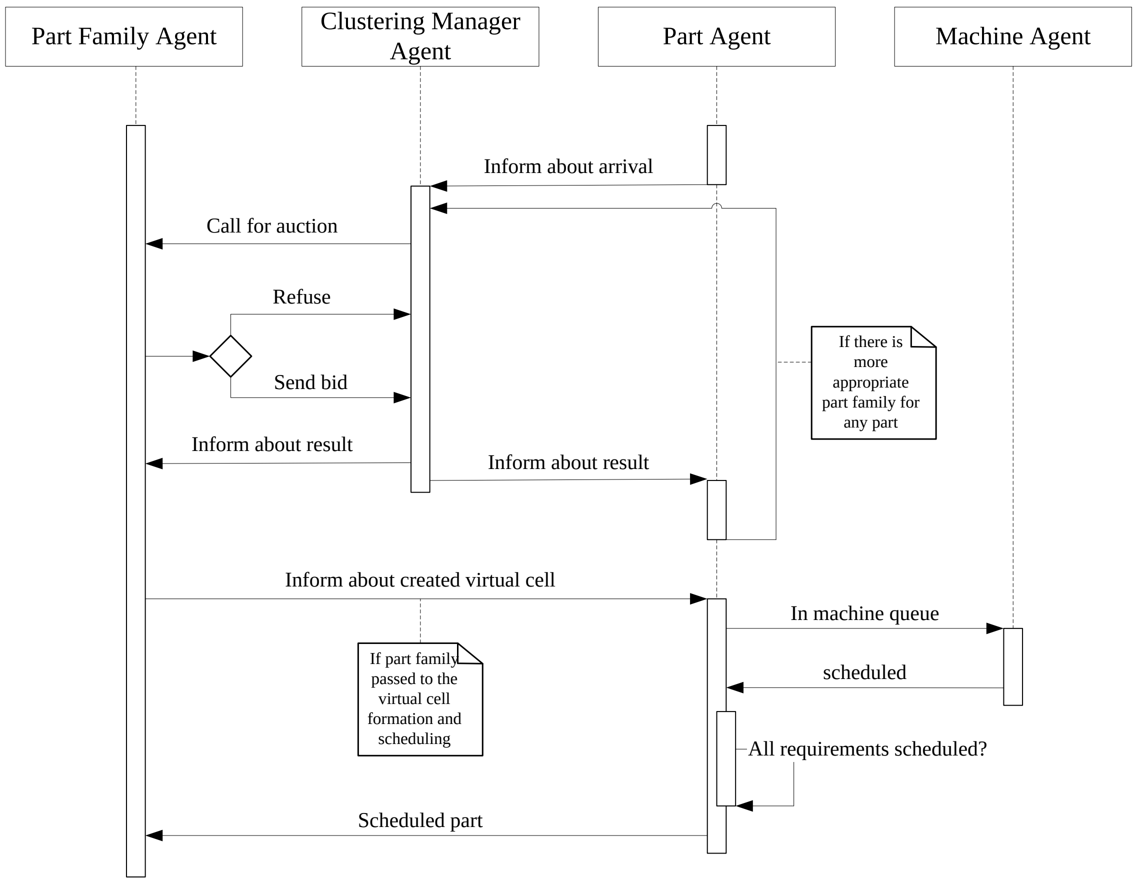

Figure 4.7 The interaction diagram of the agents

CHAPTER 5

AGENT BASED DYNAMIC VIRTUAL CELL FORMATION AND SCHEDULING APPROACH

Dynamic virtual cellular manufacturing methodology is used in order to provide an efficient manufacturing system from the arrival of the part demand to the factory to obtain finished parts. Four types of agents are defined in the algorithm. These agents and their basic features and roles are as follows.

A part agent represents an individual part demand. It aims to find the most appropriate part family for itself before the scheduling process starts. By the time the scheduling process starts, it is scheduled in the virtual cell which is determined by its part family. Each part agent has information on its arrival time, processing sequences, due date, lot size etc. The part agent represents the part demand and its all properties.

A part family agent works to form a part family which has similar parts in terms of manufacturing features and due dates. Each part family agent always makes controls to find the right time to pass from the part family formation phase to the scheduling phase. Part family agents which are in the scheduling phase determine virtual cells by evaluating the machines and constraints in the manufacturing system. Each part family agent has information about the parts which belong to it, the threshold value for acceptance of parts, etc.

Each machine agent has information about its capabilities, the process times of its capabilities, busy times, etc. These agents communicate with the parts during the scheduling process.

The clustering manager agent coordinates the part family formation phase. It communicates with parts and part families in order to arrange efficient part families.

The details of the part family formation phase are presented in the following.

5.1. Part family formation phase

The aim of this phase is to obtain part families which consist of similar parts in terms of their production features as well as their due dates. As parts enter the system dynamically, we can release the parts to the job shop immediately or use an order review/release mechanism. Sabuncuoglu and Karapinar (1999) made a study on order review/release mechanisms in production systems. They emphasised that although in most studies order review/release activities are ignored, in practice demands are often collected in a pool and then released to the manufacturing system according to a specific criterion. They stated that the target of order review/release mechanisms is to improve production system performance by controlling the input of production orders to the system. In the proposed approach we used the order review/release mechanism in order to model the manufacturing system as in real life and to take advantage of order review mechanisms. During this waiting time period, each part which is in the pool has the opportunity to find a more appropriate part family. The part family formation phase is carried out between the parts and part families when they are in the pool, that is, before releasing part families as parts for manufacturing. By the time the part families are released for manufacturing, the part family formation phase finishes and the virtual cell formation and scheduling phase starts for them. Part family formation is realised as in Figure 5.1.

```
1.1 Part agent → New part arrival
    Send a message to the clustering manager agent to inform the
    manager about the arrival.
1.2 Clustering manager agent → Opening an auction
    Put the part/parts to the Auction List
    Send a message to the part families for calling them to the
    auction for the part/parts which apply for bidding out
1.3 Part family agent → Joining to the auction
    for each part p in the Auction List
       Determine a bid for part p (bid quantity for part p= average
       dissimilarity of part p) if the part p satisfies the listed
       conditions
          The average dissimilarity of part p<=threshold of the part
          family
          The average dissimilarity of the part<average
          dissimilarity current part family (if there is)
          Entrance of part p does not make the parts in the part
          family late
          In this auction period, if part p does not present in the
          part family twice
    Select part p (among the parts in the Auction List) which has
    the minimum determined bid for joining to the auction
1.4 Clustering manager agent → Determining the winner part and part
    family
    if there are any bids
       Evaluate all the bids and find the minimum bid
       Determine the winner part family which gives the minimum bid,
       and determine the winner part to which the minimum bid is
       given
    else
       for each part p in the Auction List
          if (the average dissimilarity of the part p<average
          dissimilarity current part family)|| (part p has no
          family)
             Create a part family for part p
1.5 if there are any changes in the part families go to step 1.6,
    else finish the auction
1.6 Part agent → Looking for a new part family
    Apply to the clustering manager agent if any part family
    satisfies at least one the following conditions
       There is a nearer part family than its current part family
       Not being within the threshold values of the current part
       family
       There is only one part in the current part family
1.7 If there is any application to the clustering manager go to step
    1.2, else finish the auction period
```

Figure 5.1 Agent based dynamic part family formation algorithm

In the part family formation algorithm the average dissimilarity ADS_{if} between part i and part family f having s parts is calculated by Equation 5.1.

$$ADS_{if} = \frac{\sum_{j=1}^{s} DS_{ij}}{s} \tag{5.1}$$

where DS_{ij} is the overall dissimilarity level which is used by Baykasoglu and Gindy (2000). It considers commonality in machine requirements and similarity patterns of production sequences. The overall dissimilarity level is calculated between part *i* and part *j* by Equation 5.2 (Baykasoglu and Gindy, 2000).

$$DS_{ij} = w_1 * PDS_{ij} + w_2 * SDS_{ij} \tag{5.2}$$

w_1 and w_2 are weights on each dissimilarity index. PDS_{ij}, which is calculated by Equation 5.3, defines the part dissimilarity based on commonality of machine requirements (Baykasoglu and Gindy, 2000).

$$PDS_{ij} = 1 - (P_i \cap P_j) / (P_i \cup P_j) \tag{5.3}$$

where P_i and P_j are the operation sequences of part *i* and part *j* respectively. Here, the numerator illustrates the common operations between part *i* and part *j*, and the denominator shows the total number of operations of part *i* and part *j*.

SDS_{ij} in Equation 5.2 indicates part dissimilarity by considering the processing sequences of parts. A dynamic programming procedure is used, as in the study of Baykasoglu and Gindy (2000). This procedure was proposed by Tam (1990) for determining part dissimilarity based on the processing sequences of parts. It is given in Figure 5.2 (Tam, 1990).

```
Set M[0,0]=0
Set the first row as (M[0,k], 0<=k<=m)
Set the first column as (M[r,0], 0<=r<=n)
for (k=1 to n)
   for (r=1 to m)
      if (Pi(k)==Pj(r))
         substitude =M[k-1,r-1]
      else
         substitude=M[k-1,r-1]+1
      delete=M[k-1,r]+1
      addition=M[k,r-1]+1
      M[k,r]=min(substitude,delete,addition)
SDSi,j=M[n,m]
```

Figure 5.2 The procedure for determining part dissimilarity based on processing sequences of parts

where M is an $n*m$ matrix, and the operation sequences of part *i* and part *j* are $P_i=\{O_1,O_2,O_3 \ldots O_n\}$ and $P_j=\{O_1,O_2,O_3 \ldots O_m\}$, respectively.

Besides part similarity in the operational manner, we also pay attention to similarity in due dates. This is because grouping dissimilar parts in terms of due dates can cause undesirable deviations from both the goals of efficient manufacturing and customer satisfaction. In the part family formation algorithm, grouping of similar parts by considering due dates is provided by accepting the new part which does not make the parts in the part family late by its arrival to the part family. The parts in the part family, including the new part, are sorted according to the job scheduling rule and calculations are realised for the parts by this order. If the calculated value by subtracting the release time of the part from the current time is smaller than zero for any part, then the part family does not accept the new part. Infinite loading, which is one of the methods used for release time determination in the literature, calculates the release time by subtracting the expected flow time from the due date of the part (Sabuncuoglu and Karapınar, 1999). The release time of each part is calculated by Equation 5.4 (given in Figure 5.3) which is based on infinite loading.

By the time the part families are released for manufacturing, the part family formation phase finishes and the virtual cell formation and scheduling phase starts for them. Details of the virtual cell formation and scheduling phase are given in Section 5.2.

5.2. Virtual cell formation and scheduling phase

As seen the parts, part families and clustering manager communicate with each other and realise the part family formation algorithm continuously in order to obtain more similar part families. Part families and their parts are present in this phase until the time of virtual cell formation and scheduling. Part families always control whether any of their parts is late or not. The transition between dynamic part family formation and dynamic virtual cell formation and scheduling is provided by this control. Each part family determines the time for passing to the virtual cell formation and scheduling phase by the following procedure given in Figure 5.3.

```
Sort the parts in the part family according to the job scheduling
rule
for(i=0; i<number of parts in the part family; i++)
   Calculate the release time of ith part by using Equation (5.4)
```

$$RT_i = DDE_i - \sum_{j=1}^{i} FTE_j \quad (5.4)$$

```
   if (time>=RTi)
      Add the ith part to the list
if (list size>0)
   Pass to dynamic virtual cell formation and scheduling phase
   Determine the capacity point as the due date of ith part which
   has the minimum due date in the list
```

Figure 5.3 The procedure for transition from dynamic part family formation to dynamic virtual cell formation and scheduling

RT_i, DDE_i, and FTE_j represent release time, due date and flow time estimation of *ith* part, respectively. According to this procedure, if a part family determines its part as late or on time for scheduling, it passes to the virtual cell formation and scheduling phase. Part families determine their virtual cells according to the entering order to the virtual cell formation phase. The capacity of the virtual cell is calculated by considering the total available capacities of machines in the virtual cell in terms of resource elements, as in the study of Baykasoglu and Gindy (2000). Each part family determines the machines for its virtual cell according to the determined capacity point and parts in the machine queues. Then cell capacity estimation of the virtual cell is determined by multiplying the virtual cell capacity by the cell capacity estimation parameter. If there is more than one virtual cell with enough capacity, then the part family selects one of them as its virtual cell. We consider two strategies here. It is expected that the travelling distance of parts is lower when the first strategy is used and that overlapping between cells is lower when the second strategy is used. In the computational study, we addressed the proposed approach by considering the first and second strategies as ABVCM-1 and ABVCM-2, respectively, and the results are discussed. The machines of the virtual cells are determined according to the procedure in Figure 5.4.

```
Determine all the machine combinations consisting of all the
required REs
for each combination
   Determine total available capacity in terms of each of the REs by
   considering the time slot (capacity point-(current time+(average
   process time of REs of the machine*parts in queue))
   Calculate cell capacity estimation in terms of REs by multiplying
   the total available virtual cell capacity by cell capacity
   estimation parameter
   Add the virtual cell which has enough capacity by considering the
   required REs to the list
if there are virtual cells with enough estimated capacity in the
list
   if the first strategy is used
      Determine virtual cell which has min distance between machines
      as the virtual cell of the part family
   else if the second strategy is used
      Determine virtual cell which has min total queue as the
      virtual cell of the part family. If a tie occurs the virtual
      cell which has min distance between machines is selected
else
   Determine virtual cell which has max capacity as the virtual cell
   of the part family
```

Figure 5.4 The procedure for determination of the machines of the virtual cell

After the creation of the virtual cell, part families are scheduled. Each part is scheduled by considering machines in its virtual cell according to the scheduling rule. Overlapping between part families can occur. If there is more than one machine available for the resource element of the part in its virtual cell, then the part enters the queue of the machine which is determined using the machine selection rule.

CHAPTER 6

COMPUTATIONAL STUDY

The dynamic agent based virtual cell formation and scheduling model was developed using the multi-method simulation software AnyLogic^{R}. For the evaluation of the proposed approach, an example based on a case was prepared. In the example, the basic characteristics of the manufacturing environment, such as the processing times of resource elements, travelling distance between machines, and operation sequences of each part type, are gathered from the studies presented by Baykasoğlu (1999), Baykasoğlu (2003), and Baykasoğlu and Göçken (2010) with some assumptions. In the manufacturing environment there are 24 machines, each defined with its capability based resource elements.

The properties of the demand are determined dynamically by its arrival time. Any operation of each part can be processed on any machine which has the capability to process the required resource element. The processing capabilities of machines in terms of resource elements are given in Table 6.1 (Baykasoğlu, 2003 and Baykasoğlu and Göçken, 2010). The processing time of resource elements considering each machine type is given in Table 6.2 (Baykasoğlu and Göçken, 2010). At any given time, single resource element can be processed on a machine and preemption of an operation and/or a lot is not allowed.

Table 6.1 Processing capabilities of machines in terms of resource elements (Baykasoğlu, 2003 and Baykasoğlu and Göçken, 2010)

	Machines																						
REs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
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																		✔						

Table 6.2 Processing times of resource elements with respect to machines (minutes) (Baykasoğlu and Göçken, 2010)

REs	Machines																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1									5.8	5.8										5.80	5.10			
2									8.11	8.12										8.10	8.12			
3	6.90																				6.90			6.90
4									8.13	8.12										8.11	8.11			
5											7.13	7.13												
6									5.90	5.90										5.90	5.90			
7									9.13	9.13										9.13	9.13			
8						6.12	6.12	6.12																
9			5.90			5.90	5.90	5.90																
10			7.90																					
11						9.12	9.13	9.12																
12													6.11											
13						5.10	5.10	5.10																
14			5.70																					
15			6.70																					
16													8.13											
17														6.11										
18		6.90													6.10	6.10								
19		5.70													10.14	10.15								
20															5.90	5.90								
21																						7.11	7.11	
22		4.60													8.12	8.11								
23		4.70													7.11	7.12								
24				6.90	6.90													6.10	6.10					
25				11.14	11.14												11.14	11.14	11.14					
26				9.11	9.11												9.12	9.14	9.11					
27				6.10	6.10													6.10	6.10					
28				6.10	6.10												6.10	6.12	6.10					
29				6.10	6.10												6.10	5.90	6.10					
30				7.10	7.10												7.10	7.10	7.10					
31				13.17	13.16												13.16	13.16	6.12					
32																		6.90						

Travelling distances between machines are given in Table 6.3 (Baykasoğlu, 2003 and Baykasoğlu and Göçken, 2010). Machines are arranged according to the capability based distributed layout method, and the distances are calculated considering rectilinear movements. Researchers can find details of the capability based distributed layout method, capability based distributed layout of the manufacturing system, and related computations in the study of Baykasoğlu (2003) and Baykasoğlu and Göçken (2010).

Table 6.3 Travelling distances between machines in the capability based distributed layout (Baykasoğlu, 2003 and Baykasoğlu and Göçken, 2010)

	Machines																							
Machines	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	-	5	4	7	2	6	5	1	2	3	1	6	4	5	7	2	6	3	4	8	5	3	4	3
2	5	-	3	2	7	1	6	4	3	6	6	3	5	4	4	5	5	4	1	3	2	2	7	8
3	4	3	-	3	4	2	3	3	2	4	3	2	2	1	3	2	2	1	2	4	1	1	4	5
4	7	2	3	-	5	1	4	6	5	4	6	1	3	2	2	5	3	4	3	1	2	4	5	6
5	2	7	4	5	-	6	3	3	4	1	1	4	2	3	5	2	4	3	6	6	5	5	2	1
6	6	1	2	1	6	-	5	5	4	5	5	2	4	3	3	4	4	3	2	2	1	3	6	7
7	5	6	3	4	3	5	-	4	3	2	4	3	1	2	2	3	1	2	5	3	4	4	1	2
8	1	4	3	6	3	5	4	-	1	2	2	5	3	4	6	1	5	2	3	7	4	2	3	4
9	2	3	2	5	4	4	3	1	-	3	3	4	2	3	5	2	4	1	2	6	3	1	4	5
10	3	6	4	4	1	5	2	2	3	-	2	3	1	2	4	1	3	2	5	7	4	4	1	2
11	1	6	3	6	1	5	4	2	3	2	-	5	3	4	6	1	5	2	5	7	4	4	3	2
12	6	3	2	1	4	2	3	5	4	3	5	-	2	1	1	4	2	3	2	2	1	3	4	5
13	4	5	2	3	2	4	1	3	2	1	3	2	-	1	3	2	2	1	4	4	3	3	2	3
14	5	4	1	2	3	3	2	4	3	2	4	1	1	-	2	3	1	2	3	3	2	2	3	4
15	7	4	3	2	5	3	2	6	5	4	6	1	3	2	-	5	1	4	3	1	2	4	3	4
16	2	5	2	5	2	4	3	1	2	1	1	4	2	3	5	-	4	1	4	6	3	3	2	3
17	6	5	2	3	4	4	1	5	4	3	5	2	2	1	1	4	-	3	4	2	3	3	2	3
18	3	4	1	4	3	3	2	2	1	2	2	3	1	2	4	1	3	-	3	5	2	2	3	4
19	4	1	2	3	6	2	5	3	2	5	5	2	4	3	3	4	4	3	-	4	1	1	6	7
20	8	3	4	1	6	2	3	7	6	7	7	2	4	3	1	6	2	5	4	-	3	5	4	5
21	5	2	1	2	5	1	4	4	3	4	4	1	3	2	2	3	3	2	1	3	-	2	5	6
22	3	2	1	4	5	3	4	2	1	4	4	3	3	2	4	3	3	2	1	5	2	-	5	6
23	4	7	4	5	2	6	1	3	4	1	3	4	2	3	3	2	2	3	6	4	5	5	-	1
24	3	8	5	6	1	7	2	4	5	2	2	5	3	4	4	3	3	4	7	5	6	6	1	-

There are 20 common part types in the manufacturing system. The resource element based operation sequences of each part type are given in Table 6.4 (Baykasoğlu, 1999). However, in the system, new part demands also occur. If a new part demand occurs, its resource element based operation sequence is created randomly considering 32 resource elements. The maximum number of operations is determined as 5.

Table 6.4 RE-based operation sequences of each part type (Baykasoğlu, 1999)

Part types	Number of operations based on REs	RE based operation sequence	Part types	Number of operations based on REs	RE based operation sequence
Part 1	3	29 26 30	Part 11	4	17 18 19 21
Part 2	3	7 6 5	Part 12	3	17 19 21
Part 3	2	1 5	Part 13	3	19 21 20
Part 4	2	17 21	Part 14	2	6 7
Part 5	3	1 5 2	Part 15	2	22 21
Part 6	5	10 12 14 8 15	Part 16	3	29 24 25
Part 7	4	9 8 11 12	Part 17	4	10 8 9 12
Part 8	3	25 32 26	Part 18	4	29 24 25 32
Part 9	4	10 8 9 12	Part 19	3	1 5 3
Part 10	4	16 10 12 14	Part 20	2	26 32

Dynamically created and scheduled part families are illustrated in Figure 6.1. In order to see the change in virtual cells, three shots are presented considering different times. In Figure 6.1, when the number of part families with same virtual cells get increase, the lines of virtual cells get bolder.

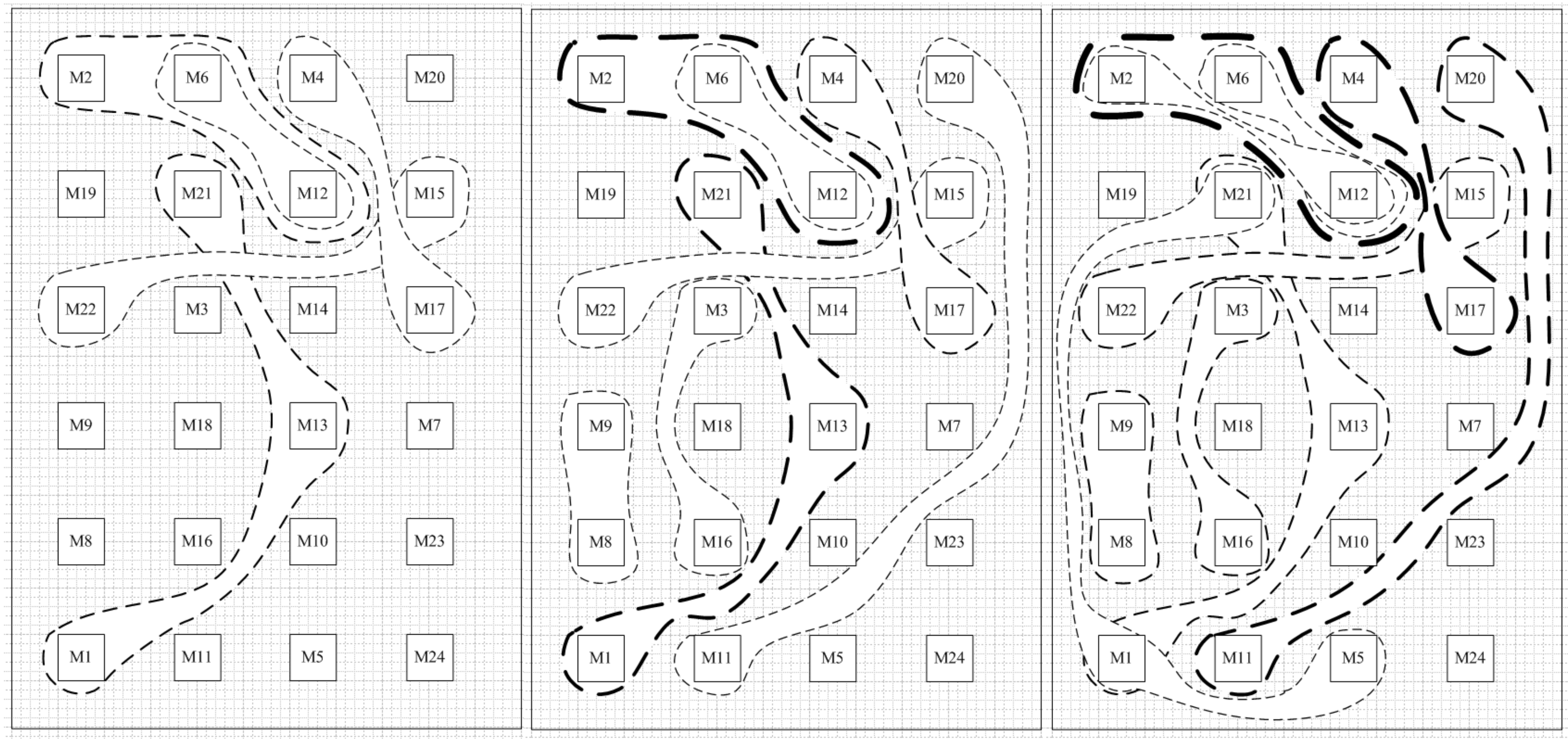

Figure 6.1 Dynamically created and scheduled part families

Parameters and their effects on the performance of the proposed algorithm and the performance of the algorithm are analysed by considering the manufacturing system which is defined above. The analyses are divided into three parts. In the first part, the parameters which directly affect the performance of part family formation are examined. In the second, the part scheduling rules are investigated. In the third part, the performance of the proposed algorithm is compared with those of functional job shop. In the analysis results of each experiment is presented for 3000 finished parts (150 lots), and lot size is considered as 20 parts. Due dates and flow time estimations of the parts are calculated according to total work content rule, which is one the most commonly used rules in the literature.

In the manufacturing system two types of set up time are considered. One is minor set up time, which occurs when a machine changes working the current resource element. It is calculated by multiplying the processing time of the lot of the part in the related machine by the minor set up time ratio. The other is major set up time, which occurs when a machine changes the part family for which it is working. Major set up time is calculated for each machine by multiplying the average processing time of the lot of the part in the machine by the major set up time ratio.

A Taguchi experimental design was prepared in the statistical software Minitab^{R} by considering the parameters and their levels which directly affect the part family formation. These parameters and their levels are given in Table 6.5. In this part of the analyses, major set up time, minor set up time, job selection rule, machine selection rule, and new part arrival rate are taken as 0.2, 0.01, earliest due date (EDD), minimum queue length based (MQLB), and 0.1, respectively. Demand arrival rate is considered as EXPO(10). The value of weight w_1 and w_2 on each dissimilarity index is taken as 0.5.

Table 6.5 Parameters and their levels

Parameters	Levels
Threshold (t) of part family	0.5, 1.0, 1.5
Cell capacity estimation parameter(cce)	0.5, 0.75, 1.0
Flow time estimation parameter (fte)	2.0, 3.0
Due date estimation parameter (dde)	6.0, 8.0, 10.0

The results of the each experiment are presented in Table 6.6 (average of two runs). The average tardiness and average set up time performance measures are selected for the evaluation. This is because these measures can be directly affected by the considered parameters. Also, manufacturers need to meet the demand of customers on time. One of the most important reasons for preferring cellular manufacturing over job shops is the opportunity to work with minimum set up times. Therefore, determining appropriate levels for the parameters by considering average tardiness and average set up time is important.

Table 6.6. Results of experiments

Exp. no	t	cce	fte	dde	Average tardiness	Average set up time
1	0.5	0.50	2	6	69.49	55.19
2	1.0	0.75	2	6	114.72	58.32
3	1.5	1.00	2	6	134.10	62.54
4	0.5	0.50	2	8	10.90	50.26
5	1.0	0.75	2	8	21.08	47.15
6	1.5	1.00	2	8	24.34	54.33
7	0.5	0.75	2	10	2.65	43.25
8	1.0	1.00	2	10	7.51	42.52
9	1.5	0.50	2	10	2.95	44.47
10	0.5	1.00	3	6	93.22	59.84
11	1.0	0.50	3	6	85.00	62.42
12	1.5	0.75	3	6	132.44	74.28
13	0.5	0.75	3	8	78.26	57.40
14	1.0	1.00	3	8	42.48	56.94
15	1.5	0.50	3	8	16.08	59.72
16	0.5	1.00	3	10	1.29	47.48
17	1.0	0.50	3	10	5.92	46.05
18	1.5	0.75	3	10	2.72	54.15

The effects of the parameters on the average tardiness values and average setup times are illustrated in Figure 6.2 and Figure 6.3, respectively.

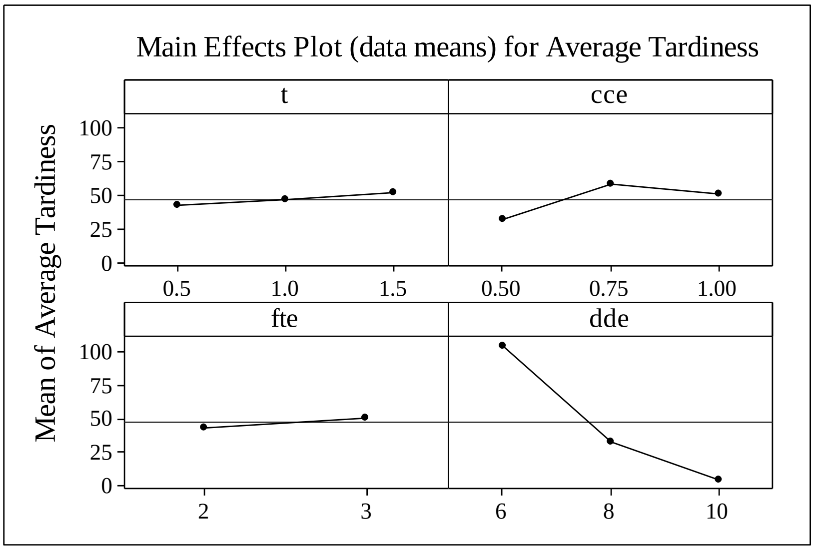

Figure 6.2 The main effects plot for average tardiness

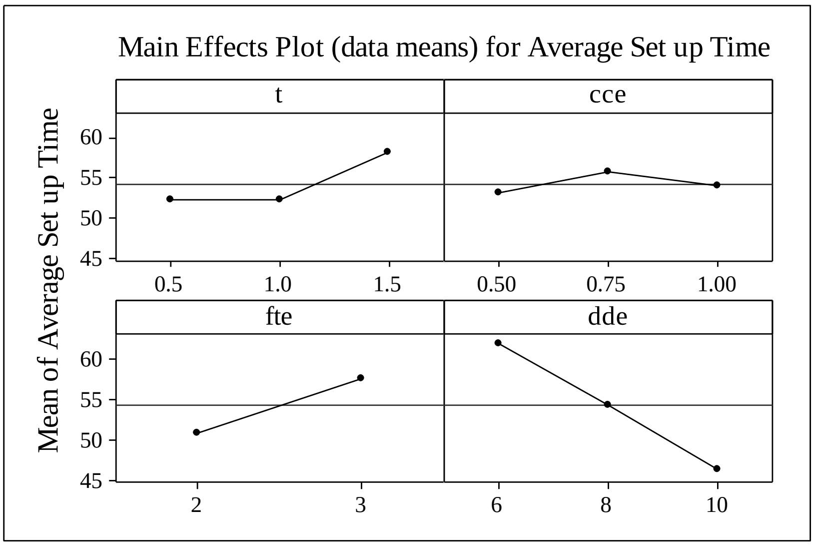

Figure 6.3 The main effects plot for average set up time

As the threshold value increases, the similarity of parts in the part family decreases and the number of parts in the part family increases. It is observed that when the threshold value is 0.5, the created part families usually consist of the same type of parts. On the other hand, when the threshold value is 1.0, part families consist of similar parts (different types of parts besides the same type of parts). In Figure 6.2 and Figure 6.3 there is no significant difference between the level 0.5 and level 1.0 in the results. A threshold level of 1.5 increases the average set up time significantly. Therefore we can say that 1.0 is the appropriate level for the threshold parameter. One of the important parameters is the cell capacity estimation parameter. This parameter affects the determination of sufficient capacity to the part families. If the capacity is estimated incorrectly, deviations will be larger from the due dates. Figures 6.2 and Figure 6.3 show that the level 0.5 is appropriate for the cell capacity estimation parameter considering both average tardiness and average set up time. Flow time estimation and the due date estimation parameter can be evaluated together. This is because when these parameter values get closer, the time for part family formation gets lower. Also, according to the proposed approach one of the constraints for joining a family is that accepting the part to the part family should not make the current parts in the part family late. Therefore, we can expect that grouping can be realised by considering longer times and parts when the values of flow time and due date estimation parameters are 2 and 10, respectively. We see in the results that these levels are the most desired ones in terms of average set up time and average tardiness performance measurements. Therefore these levels are considered in the following analyses.

The average time in shop and average tardiness performance measurements are considered in the determination of the appropriate scheduling rules, since these measurements will be greatly affected by these rules. The earliest due date and shortest process time rules are considered as the part scheduling rules and the minimum queue length based and minimum load based rules are considered as the machine selection rules. The results are given in Table 6.7 and illustrated in Figure 6.4 and Figure 6.5.

Table 6.7 Results according to scheduling rules

Part selection rule	Machine selection rule	Average time in shop	Average tardiness
EDD	MQLB	1427.90	2.45
SPT	MQLB	1464.18	11.39
EDD	MLB	1463.82	4.20
SPT	MLB	1690.89	56.62

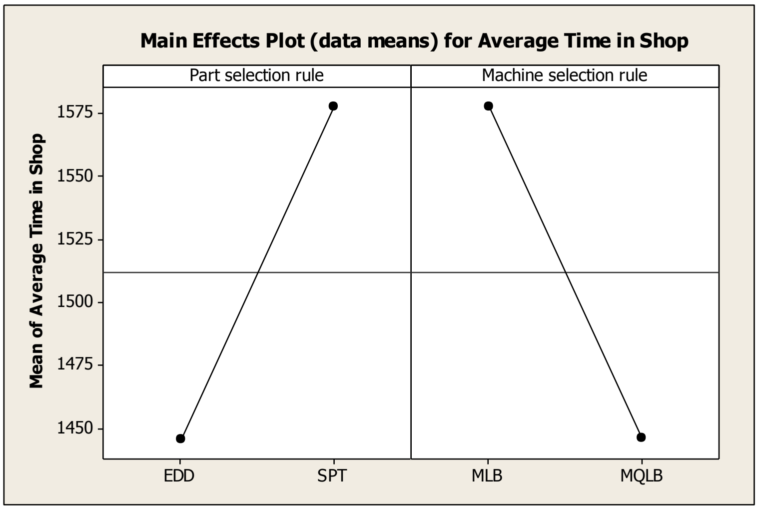

Figure 6.4 Main effects plot considering average time in shop

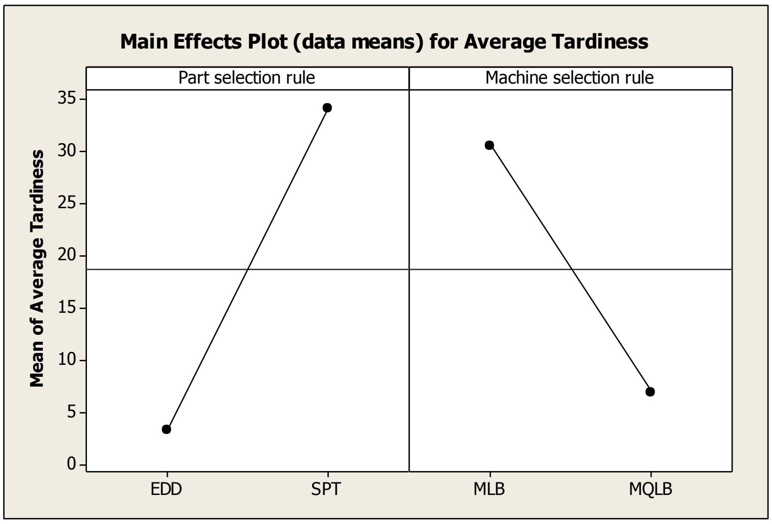

Figure 6.5 Main effects plot considering average tardiness

Results show that the EDD part scheduling rule and minimum queue length based machine selection rule are significantly preferable. These scheduling rules are considered in the comparisons.

We compared the results of the proposed approach with the results of a functional job shop. In the functional manufacturing system, the same scheduling rules, namely EDD and MQLB, are used as the part scheduling rule and the machine selection rules, respectively. The strategy for the part demand arrivals is the same as with the proposed approach. Minor set up time and major set up time occur in the machine with the change of the processing resource element and part, respectively. Travelling distances between machines in the functional layout is given in Table 6.8 (Baykasoğlu and Göçken 2010).

Table 6.8 Travelling distances between machines in functional layout (Baykasoğlu and Göçken 2010)

	Machines																							
Machines	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	-	5	4	7	2	6	5	1	2	3	1	6	4	5	7	2	6	3	4	8	5	3	4	3
2	5	-	3	2	7	1	6	4	3	6	6	3	5	4	4	5	5	4	1	3	2	2	7	8
3	4	3	-	3	4	2	3	3	2	4	3	2	2	1	3	2	2	1	2	4	1	1	4	5
4	7	2	3	-	5	1	4	6	5	4	6	1	3	2	2	5	3	4	3	1	2	4	5	6
5	2	7	4	5	-	6	3	3	4	1	1	4	2	3	5	2	4	3	6	6	5	5	2	1
6	6	1	2	1	6	-	5	5	4	5	5	2	4	3	3	4	4	3	2	2	1	3	6	7
7	5	6	3	4	3	5	-	4	3	2	4	3	1	2	2	3	1	2	5	3	4	4	1	2
8	1	4	3	6	3	5	4	-	1	2	2	5	3	4	6	1	5	2	3	7	4	2	3	4
9	2	3	2	5	4	4	3	1	-	3	3	4	2	3	5	2	4	1	2	6	3	1	4	5
10	3	6	4	4	1	5	2	2	3	-	2	3	1	2	4	1	3	2	5	7	4	4	1	2
11	1	6	3	6	1	5	4	2	3	2	-	5	3	4	6	1	5	2	5	7	4	4	3	2
12	6	3	2	1	4	2	3	5	4	3	5	-	2	1	1	4	2	3	2	2	1	3	4	5
13	4	5	2	3	2	4	1	3	2	1	3	2	-	1	3	2	2	1	4	4	3	3	2	3
14	5	4	1	2	3	3	2	4	3	2	4	1	1	-	2	3	1	2	3	3	2	2	3	4
15	7	4	3	2	5	3	2	6	5	4	6	1	3	2	-	5	1	4	3	1	2	4	3	4
16	2	5	2	5	2	4	3	1	2	1	1	4	2	3	5	-	4	1	4	6	3	3	2	3
17	6	5	2	3	4	4	1	5	4	3	5	2	2	1	1	4	-	3	4	2	3	3	2	3
18	3	4	1	4	3	3	2	2	1	2	2	3	1	2	4	1	3	-	3	5	2	2	3	4
19	4	1	2	3	6	2	5	3	2	5	5	2	4	3	3	4	4	3	-	4	1	1	6	7
20	8	3	4	1	6	2	3	7	6	7	7	2	4	3	1	6	2	5	4	-	3	5	4	5
21	5	2	1	2	5	1	4	4	3	4	4	1	3	2	2	3	3	2	1	3	-	2	5	6
22	3	2	1	4	5	3	4	2	1	4	4	3	3	2	4	3	3	2	1	5	2	-	5	6
23	4	7	4	5	2	6	1	3	4	1	3	4	2	3	3	2	2	3	6	4	5	5	-	1
24	3	8	5	6	1	7	2	4	5	2	2	5	3	4	4	3	3	4	7	5	6	6	1	-

In order to analyse the effects of set up time, two different levels of major set up time ratio are considered. These are 0.2 and 0.6, because one of the advantages of cellular manufacturing is working with lower set up times. As known, one of the most important drawbacks of cellular manufacturing versus job shops is inefficiency in handling the new type of part demands. In order to observe the behaviour of the proposed approach by considering the varying rate of new type of part demand arrivals, we consider two levels for the new type part demand arrival rate. These levels are 0.1 and 0.3. The results considering 10 independent runs are presented in Table 6.9.

Table 6.9 Summary of results

Methods	Major setup time ratio	New part arrival rate	Time in shop			Set up time			Total travel time		
			Min	Ave	Max	Min	Ave	Max	Min	Ave	Max
ABVCM-1	0.2	0.1	1228.7	1362.1	1514.4	38.8	41.6	46.4	302.0	350.1	384.0
ABVCM-1	0.2	0.3	1025.7	1115.5	1207.8	40.9	44.1	47.1	298.0	329.5	386.0
ABVCM-1	0.6	0.1	1458.1	1573.4	1745.9	116.3	126.7	139.9	305.0	360.9	416.0
ABVCM-1	0.6	0.3	1255.6	1327.1	1399.5	54.7	126.5	142.9	280.0	325.6	390.0
ABVCM-2	0.2	0.1	1139.8	1243.1	1341.8	34.6	40.0	44.5	393.0	453.1	591.0
ABVCM-2	0.2	0.3	927.0	996.6	1055.6	36.4	40.6	45.1	348.0	385.8	423.0
ABVCM-2	0.6	0.1	1363.1	1569.4	1729.3	106.9	125.4	140.9	397.0	451.2	562.0
ABVCM-2	0.6	0.3	1109.7	1190.2	1287.7	108.6	117.7	126.0	368.0	408.3	471.0
Functional job shop	0.2	0.1	1029.1	1147.7	1258.3	72.2	73.9	75.5	517.0	570.9	658.0
Functional job shop	0.2	0.3	856.6	961.7	1077.8	66.0	69.4	72.6	468.0	566.6	621.0
Functional job shop	0.6	0.1	1620.8	1787.5	1902.5	208.9	217.0	224.8	479.0	567.0	634.0
Functional job shop	0.6	0.3	1189.2	1396.3	1542.1	191.6	199.6	212.1	448.0	528.8	608.0

The results of each experiment considering each solution approach in terms of average time in shop, average set up time and total travel time are illustrated in Figures 6.6, 6.7, and 6.8, respectively.

According to the results, ABVCM-1 and ABVCM-2 dominate the functional job shop in terms of average set up time and total travel time in all the experiments. Functional layout is good at average time in shop results when the major set up time ratio is 0.2, however, when this ratio is considered as 0.6, the ABVCM-1 and ABVCM-2 results are much better than those for the functional job shop. We can also say that the new type part demand arrivals can be handled by the proposed algorithm successfully. The overlapping ratio is lower in ABVCM-2 than in ABVCM-1, and it is already good at average time in shop performance measurement. ABVCM-1 outperformed ABVCM-2 in the total travel time criteria. This is not surprising, that the capacity determining strategy of ABVCM-1 supports these results.

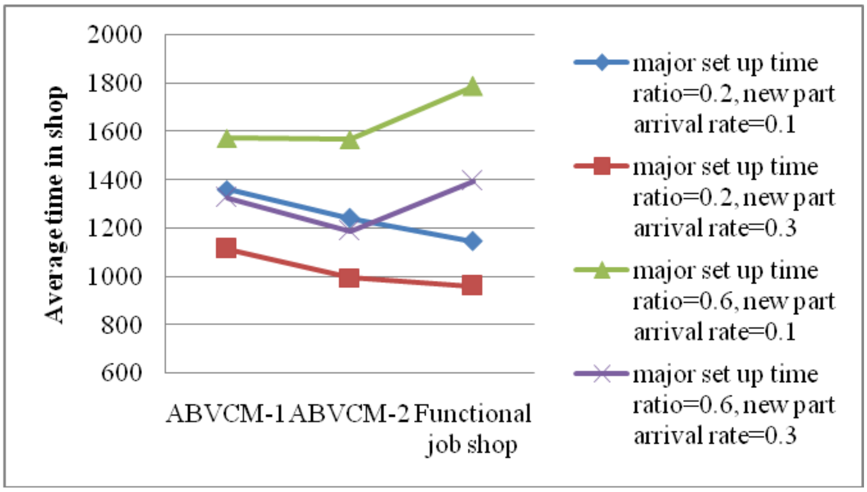

Figure 6.6 Results considering average time in shop

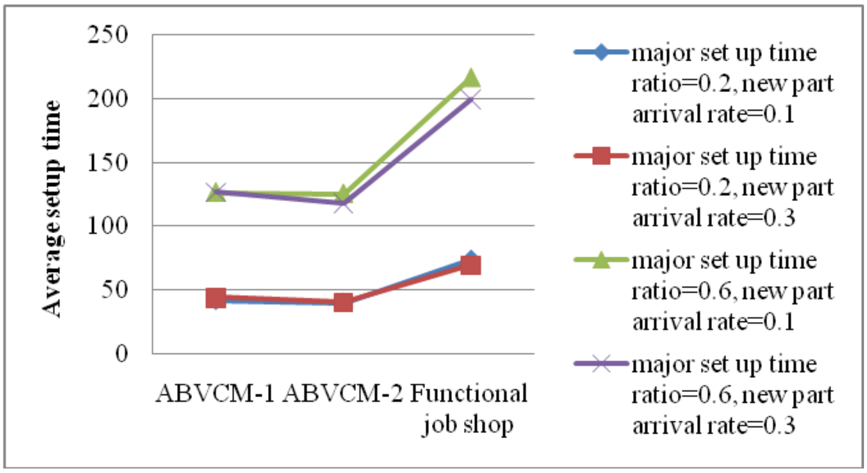

Figure 6.7 Results considering average setup time

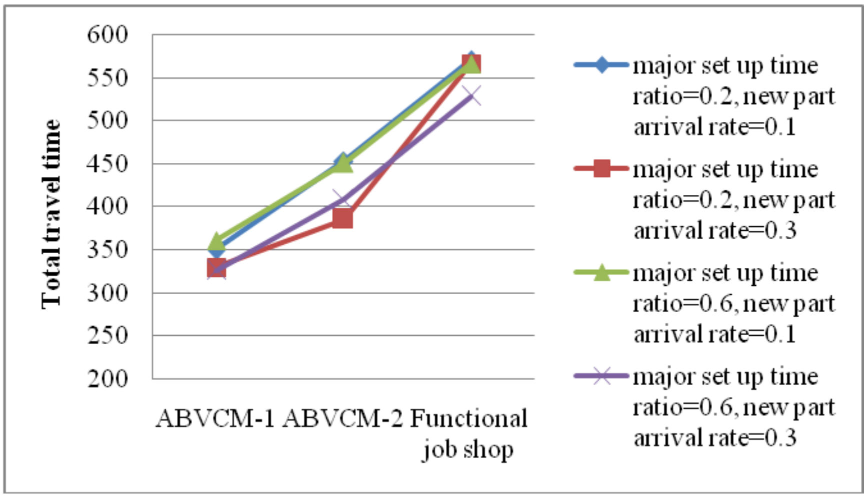

Figure 6.8 Results considering total travel time

CHAPTER 7

SUMMARY, CONCLUSIONS, AND FURTHER RESEARCH

In this chapter, a summary, the main contributions of the thesis to the literature, and the further research areas are presented.

In Chapter 1, an introduction to the research is given. And in Chapter 2, the literature review related with the topics which is considered in the thesis study is presented.

In Chapter 3, an agent based dynamic part family formation algorithm is explained. Firstly, we focus on dynamic part family formation in conceptual level. An agent based clustering algorithm for part family formation in cellular manufacturing applications is developed considering dynamic demand changes. Although the proposed algorithm is directly applicable to dynamic part family formation problems, it can also be extended to other dynamic clustering problems. Due to the unavailability of dynamic benchmark data for part family formation problems in the literature, the performance of the proposed algorithm is compared on static test problems by dynamically introducing parts in these datasets to the proposed agent based algorithm. Although the proposed agent based algorithm is not an optimization based algorithm, we have shown that it has ability to provide competitive results which are comparable to the best known solutions.

An overview of the agent dynamic agent based virtual cellular manufacturing approach is presented in Chapter 4. And the novel dynamic agent based virtual cellular manufacturing approach is explained in Chapter 5. The presented approach aims to handle dynamic part demand arrivals while providing efficiency and flexibility. It consists of several concepts and approaches to support this aim, such as agent based modelling, market oriented programming, virtual cellular manufacturing, resource elements, and capability based distributed layout. The proposed integrated methodology enables to realize part family formation, virtual cell formation, and scheduling phases simultaneously. In Chapter 6, computational study of the proposed methodology is given. The performance of the approach is tested with several experiments. The performance measurements show that the proposed approach provides promising solutions. The results also show that it has the ability to manage the dynamic part demand arrivals efficiently.

The proposed approach is very important for both industry and academia. Since manufacturers are face to face with dynamism in most of the areas. If an unpredictable event occurs, several planned issues may become meaningless. And one has to start rescheduling. And if the system is large sized and the environment is volatile, then it is becoming more diffucult. The requirement of a system which handle dynamism efficiently is detected by several researchers. And there are attempts to overcome this problem. But the efforts are not enough. Thus, we present an important study in order to fill this gap. The proposed dynamic agent based virtual cellular manufacturing system has abilities to handle dynamism in part demand arrivals and provide efficient and flexible manufacturing. It is also open for improvements. The further research areas are listed below:

1. In the dynamic virtual cellular manufacturing systems dynamic part demand arrivals are considered. Since, it is one of the most important one among the dynamics in the manufacturing environment. But the environmental dynamics can be modelled in order to obtain more realistic solutions. Algorithms to handle these dynamics can be easily adapted to the proposed approach. Since agent based modelling gives the opportunity to maintain or change the system in an easier way.

2. One of the most important mechanisms of the proposed algorithm is transition of part families from part family formation phase to virtual cell formation and scheduling phase. We used an order review mechanism which mainly considers the due dates and flow times of the parts. Besides these, capacity of the manufacturing environments can be considered. Thus, other types of order review mechanisms can be used or developed in order get a more efficient system.

3. In part family formation phase of the algorithm, agents communicate each other to obtain more similar part families in terms of manufacturing requirements and due dates. They mainly used an auction based communicating mechanism which is

popularly used as the communication mechanism of most of the agent based algorithms. New communication mechanisms can be developed.

4. In virtual cell formation and scheduling phase part family agents can investigate the manufacturing environment considering various objectives in order to create more efficient virtual cells and schedule these cells. In most of the phases of the approach the proposed algorithm can be supported by the heuristics and metaheuristics to obtain more efficient results with lower computational times.

REFERENCES

Aggarwal, C. C., Han, J., Wang, J., Yu, P. S. (2003). A framework for clustering evolving data streams, VLDB’03 Proceedings of the 29th International Conference on Very Large Data Bases, Berlin, Germany.

Ah kioon, S., Bulgak, A. A., Bektas, T. (2009). Integrated cellular manufacturing systems design with production planning and dynamic system reconfiguration, *European Journal of Operation Research*, **192**, 414-428.

Al-Sultan, K. S., Fedjkı, C. A. (1997). A genetic algorithm for the part family formation problem, *Production Planning and Control*, **8(8)**, 788-796.

AnyLogic^{R}. Available at: http://www.anylogic.com/upload/Big%20Book%20of%20AnyLogic/Designing_state-based_behavior-statecharts.pdf. Accessed by 29.11.2013

AnyLogic^{R}. Available at: http://www.anylogic.com/overview. Accessed by 08.05.2013.

AnyLogic^{R}. Available at: http://www.anylogic.com/agent-based-modeling. Accessed by 08.05.2013.

Askin, R. G., Subramanian, S. (1987). A cost-based heuristic for group technology configuration, *International Journal of Production Research*, **25**, 101-113.

Balakrishnan, J., Cheng, C. H. (2005). Dynamic cellular manufacturing under multiperiod planning horizons, *Journal of Manufacturing Technology Management*, **16 (5)**, 516-530.

Balakrishnan, J., Cheng, C. H. (2007). Multi-period planning and uncertainty issues in cellular manufacturing: a review and future directions, *European Journal of Operation Research*, **177**, 281-309.

Baykasoglu, A., Saad, S. M., Gindy, N. (1998a). A loading approach for cellular manufacturing systems, FAIM'1998: 8th International Conference on Flexible Automation and Intelligent Manufacturing, Portland, Oregon, USA, July 1-3.

Baykasoglu, A., Gindy, N. N. Z., Saad, S. M. (1998b). A framework for the reconfiguration of cellular manufacturing systems, IMS-98, 2nd Int. Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey, August 6-7.

Baykasoğlu, A. (1999). Multiple objective decision support framework for configuring, loading and reconfiguring manufacturing cells, PhD diss., University of Nottingham.

Baykasoglu, A., Gindy, N. (2000). MOCACEF 1.0: Capability based approach to form part-machine groups for cellular manufacturing applications, *International Journal of Production Research*, **38(5)**, 1133-116.

Baykasoglu, A., Gindy, N. N. Z., Cobb, R. C. (2001). Capability based formulation and solution of multiple objective cell formation problems using simulated annealing, *Integrated Manufacturing Systems: The International Journal of Manufacturing Technology Management*, **12(4)**, 258-274.

Baykasoglu, A. (2003). Capability-based distributed layout approach for virtual manufacturing cells, *International Journal of Production Research*, **41(11)**, 2597-2618.

Baykasoglu, A. (2004). A meta-heuristic algorithm to solve quadratic assignment formulations of cell formation problems without presetting number of cells, *Journal of Intelligent Manufacturing*, **15(6)**, 753-759.

Baykasoğlu, A., Göçken, M. (2010). Capability-based distributed layout and its simulation based analyses, *Journal of Intelligent Manufacturing*, **21**, 471-485.

Baykasoglu, A., Kaplanoglu, V., Erol, R., Sahin, C. (2011). A multi-agent framework for load consolidation in logistics, *Transport*, **26(3)**, 320-328.

Ben-Arieh, D., Sreenivasan, R. (1999). Information analysis in a distributed dynamic group technology method, *International Journal of Production Economics*, **60(61)**, 427-432.

Benjafaar, S., Sheikhzadeh, M. (2000). Design of flexible plant layouts, *IIE Transactions*, **32**, 309–322.

Bianchi, L. (2000). Notes on dynamic vehicle routing-the state of the art, Techical Report-IDSIA-05-01, December 20.

Boctor, F. (1991). A linear formulation of the machine-part cell formation problem, *International Journal of Production Research*, **29(2)**, 343-356.

Boe, W., Cheng, C. H. (1991). A close neighbor algorithm for designing cellular manufacturing systems, *International Journal of Production Research*, **29(10)**, 2097-2116.

Borshchev, A., Filippov, A. (2004). From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools, The 22nd International Conference of the System Dynamics Society, Oxford, England, July 25 – 29.

Branke, J. (2001). Evolutionary optimization in dynamic environments. Kluwer Academic Publishers.

Carrie, S. (1973). Numerical taxonomy applied to group technology and plant layout, *International Journal of Production Research*, **11**, 399-416.

Chan, H. M., Milner, D. A. (1982). Direct clustering algorithm for group formation in cellular manufacture, *Journal of Manufacturing System*, **1**, 65-75.

Chandrashekharan, M.P., Rajagopalan, R. (1986a). An ideal seed non-hierarchical clustering algorithm for cellular manufacturing, *International Journal of Production Research*, **24(2)**, 451-464.

Chandrashekharan, M. P., Rajagopalan, R. (1986b). MODROC: An extension of rank order clustering for group technology, *International Journal of Production Research*, **24(5)**, 1221-1233.

Chandrasekharan, M. P., Rajagopalan, R. (1987). ZODIAC-An algorithm for concurrent formation of part-families and machine-cells, *International Journal of Production Research*, **25(6)**, 835-850.

Chandrasekharan, M. P., Rajagopalan, R. (1989). Groupability: analysis of the properties of binary data matrices for group technology, *International Journal of Production Research*, **27(6)**, 1035–1052.

Chen, M. (1998). A mathematical programming model for system reconfiguration in a dynamic cellular manufacturing environment, *Annals of Operations Research*, **77**, 109-128.

Cheng, C. H., Gupta, Y. P., Lee, W. H., Wong, K. F. (1998). A tsp-based heuristic for forming machine groups and part families, *International Journal of Production Research*, **36**, 1325-1337.

Das, K., Abdul-Kader, W. (2011). Consideration of dynamic changes in machine reliability and part demand: a cellular manufacturing systems design model, *International Journal of Production Research*, **49(7)**, 2123-2142.

Davidsson, P., S. Johansson, J., Persson, J. A., Wernstedt, F. (2003). Agent-based approaches and classical optimization techniques for dynamic distributed resource allocation: a preliminary study, In AAMAS'03 workshop on Representations and Approaches for Time Critical Decentralized Resource/Role/Task Allocation.

Dimopoulos, C., and Mort, N. A. (2001). Hierarchical clustering methodology based on genetic programming for the solution of simple cell-formation problems, *International Journal of Production Research*, **39(1)**, 1-19.

Drolet, J. R. (1989). Scheduling virtual cellular manufacturing systems, PhD diss., Purdue University.

Erol, R., Sahin, C., Baykasoglu, A., Kaplanoglu, V. (2012). A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems, *Applied Soft Computing*, **12(6)**, 1720-1732.

Flower, D. (2005). A survey of market oriented programming.

Fournier, D., Simon, G., Mermet, B. (2007). A dynamic clustering algorithm for mobile objects, *Lecture Notes in Computer Science*, **4702**, 422-429.

Garro, A., Russo, W. (2010). *easyABMS*: a domain-expert oriented methodology for agent-based modeling and simulation, *Simulation Modeling Practice and Theory*, **18**, 1453-1467.

Ghotboddini, M. M, Rabbani, M., Rahimian, H. (2011). A comprehensive dynamic cell formation design: benders' decomposition approach, *Expert Systems with Applications*, **38**, 2478-2488.

Gindy, N. N. Z., Ratchev, T. M., Case, K. (1996). Component grouping for cell formation using resource elements, *International Journal of Production Research*, **34(3)**, 727–752.

Gonçalves, J. F., Resende, M. G. C. (2004). An evolutionary algorithm for manufacturing cell formation, *Computers and Industrial Engineering*, **47**, 247-273.

Hamedi, M., Esmaeilian, G. R., Ismail, N., Ariffin, M. K. A. (2012). Capability based virtual cellular manufacturing systems formation in dual resource constrained settings using tabu search, *Computers and Industrial Engineering*, **62**, 953-971.

Islier, A. A. (2005). Group technology by an ant system algorithm, *International Journal of Production Research*, **43(5)**, 913-932.

James, T., Brown, E., Keeling, K. (2007). A hybrid grouping algorithm for the cell formation problem, *Computers and Operations Research*, **34**, 2059-2079.

Kannan, V. R., Ghosh, S. (1996). Cellular manufacturing using virtual cells, *International Journal of Operations and Production Management*, **16(5)**, 99-112.

Kannan, V. R. (1997). A simulation analysis of the impact of family configuration on virtual cellular manufacturing, *Production Planning & Control: The Management of Operations*, **8(1)**, 14-24.

Kannan, V. R. (1998). Analysing the trade-off between efficiency and flexibility in cellular manufacturing systems, *Production Planning & Control: The Management of Operations*, **9(6)**, 572-579.

Karageorgos, A., Mehandjiev, N., Weichhart, G., Hammerle, A. (2003). Agent-based optimisation of logistics and production planning, *Engineering Applications of Artificial Intelligence*, 16(4), 335-348.

Kao, Y., Li, Y. L. (2008). Ant colony recognition systems for part clustering problems, *International Journal of Production Research*, **46(15)**, 4237-4258.

Keeling, K. B., Brown, E. C., James, T. L. (2007). Grouping efficiency measures and their impact on factory measures for the machine-part cell formation problem: a simulation study, *Engineering Applications of Artificial Intelligence*, **20(1)**, 63-78.

Kesen, S. E., Toksari, M. D., Güngör, Z., Güner, E. (2009). Analyzing the behaviors of virtual cells (VCs) and traditional manufacturing systems: ant colony optimization (ACO)-based metamodels, *Computers and Operations Research*, **36**, 2275-2285.

Kesen, S. E., Das, S. K., Gungor, Z. (2010a). A mixed integer programming formulation for scheduling of virtual manufacturing cells (VMCs), *The International Journal of Advanced Manufacturing Technology*, **47**, 665-678.

Kesen, S. E., Das, S. K., Güngör, Z. (2010b). A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (VMCs), *Computers & Operations Research*, **37**, 1148-1156.

Kesen, S. E., Güngör, Z. (2012). Job scheduling in virtual manufacturing cells with lot-streaming strategy: a new mathematical model formulation and a genetic algorithm approach, *Journal of the Operational Research Society*, **63**, 683-695.

Khalilian, M., Mustapha, N. (2010). Data stream clustering: challenges and issues, Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, March 17-19.

Khilwani, N., Ulutas, B. H., Islier, A. A., Tiwari, M. K. (2011). A methodology to design virtual cellular manufacturing systems, *Journal of Intelligent Manufacturing*, **22**, 533-544.

King, J. R., Nakornchai, V. (1982). Machine-component group formation in group technology: review and extension, *International Journal of Production Research*, **20(2)**, 117-133.

King, J. R. (1980). Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm, *International Journal of Production Research*, **18(2)**, 213-232.

Kiselev, I., Alhajj, R. (2008). A self-organizing multi-agent system for online unsupervised learning in complex dynamic environments, First International Workshop on: Optimization in Multi-Agent Systems.

Ko, K. C., Egbelu, P. J. (2003). Virtual cell formation, *International Journal of Production Research*, **41(10)**, 2365-2389.

Kumar, K. R., Chandrasekharan, M. P. (1990). Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology, *International Journal of Production Research*, **28(2)**, 233-243.

Kumar, K. R., Kusiak, A., Vannelli, A. (1986). Grouping of parts and components in flexible manufacturing systems, *European Journal of Operations Research*, **24**, 387-397.

Kumar, K. R., Vannelli, A. (1987). Strategic subcontracting for efficient disaggregated manufacturing, *International Journal of Production Research*, **25(12)**, 1715–1728.

Kusiak, A., Cho, M. (1992). Similarity coefficient algorithm for solving the group technology problem, *International Journal of Production Research*, **30**, 2633-2646.

Kusiak, A., Chow, W. (1987). Efficient solving of the group technology problem, *Journal of Manufacturing Systems*, **6(2)**, 117-124.

Kusiak, A., Vannelli, A., Kumar, K. R. (1986). Clustering analysis: model and algorithms, *Control and Cybernetics*, **15**, 139-153.

Lee, S., Kim, G., Kim, S. (2011). Self-adaptive and dynamic clustering for online anomaly detection, *Expert Systems with Applications*, **38**, 14891-14898.

Macal, C. M., North, M. J. (2009). Agent-based modeling and simulation, Proceedings of the 2009 Winter Simulation Conference, December 13-16.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability.

McCormick, W. T., Schweitzer, P. J., White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique, *Operations Research*, **20**, 993-1009.

Megala, N., Rajendran, C., Gopalan, R. (2008). An ant colony algorithm for cell-formation in cellular manufacturing systems, *European Journal of Industrial Engineering*, **2(3)**, 298-336.

Mahdavi, I., Aalaei, A., Paydar, M. M., Solimanpur, M. (2009). Production planning and cell formation in dynamic virtual cellular manufacturing systems with worker flexibility, Computers and Industrial Engineering CIE 2009, 6-9 July.

Mahdavi, I., Aalaei, A., Paydar, M. M., Solimanpur, M. (2011). Multi-objective cell formation and production planning in dynamic virtual cellular manufacturing systems, *International Journal of Production Research*, **49(21)**, 6517-6537.

Mak, K. L., Lau, J. S. K., Wang, X. X. (2005). A genetic scheduling methodology for virtual cellular manufacturing systems: an industrial application, *International Journal of Production Research*, **43(12)**, 2423-2450.

Mak, K. L., Peng, P., Wang, X. X., Lau, T. L. (2007). An ant colony optimization algorithm for scheduling virtual cellular manufacturing systems, *International Journal of Computer Integrated Manufacturing*, **20(6)**, 524-537.

McLean, C. R., Bloom, H. M., Hopp, T. H. (1982). The virtual cell, Proceedings of 4th IFAC/IFIP Conference on Information Control Problems in Manufacturing Technology Gaithersburg, MD, October.

Mosier, C. T., Taube L. (1985a). The facets of group technology and their impact on implementation, *Omega*, **13(6)**, 381-391.

Mosier, C. T., Taube, L. (1985b). Weighted similarity measure heuristics for the group technology machine clustering problem, *Omega*, **13(6)**, 577-583.

Murali, R. V., Puri, A. B., Prabhakaran, G. (2010). Artificial neural networks based predictive model for worker assignment into virtual cells, *International Journal of Engineering, Science and Technology*, **2(1)**, 163-174.

Murali, R. V. (2012). Workforce assignment into virtual cells using learning vector quantization (LVQ) approach, *Research Journal of Applied Sciences, Engineering and Technology*, **4(15)**, 2427-2435.

Muruganandam, A., Prabhakaran, G., Murali, R. V. (2008). PRABHA - a new heuristic approach for machine cell formation under dynamic production environments, *The International Journal of Applied Management and Technology*, **6(3)**, 191-221.

Nomden, G., Zee, D. J. van der. (2008). Virtual cellular manufacturing: configuring routing flexibility, *International Journal of Production Economics*, **112**, 439-451.

Onwubolu, G., Mutingi M. (2001). A genetic algorithm approach to cellular manufacturing systems, *Computer and Industrial Engineering*, **39**, 125-144.

Papaioannou, G., Wilson, J. M. (2010). The evolution of cell formation problem methodologies based on recent studies (1997-2008): review and directions for future research, *European Journal of Operational Research*, **206**, 509-521.

Psaraftis, H. (1995). Dynamic vehicle routing: status and prospects, *Annals of Operations Research*, **61(1)**, 143–164.

Rezazadeh, H., Mahinib, R., Mahdi, Z. (2011). Solving a dynamic virtual cell formation problem by linear programming embedded particle swarm optimization algorithm, *Applied Soft Computing*, **11**, 3160-3169.

Saad, S. M., Baykasoglu, A., Gindy, N. N. Z. (2002a). A new integrated system for loading and scheduling in cellular manufacturing, *International Journal of Computer Integrated Manufacturing*, **15(1)**, 37-49.

Saad, S. M., Baykasoglu, A., Gindy, N. N. Z. (2002b). An integrated framework for reconfiguration of cellular manufacturing systems using virtual cells, *Production Planning & Control: The Management of Operations*, **13(4)**, 381-393.

Sabuncuoglu, I., Karapinar, H. Y. (1999). Analysis of order review/release problems in production systems, *International Journal of Production Economics*, **62**, 259-279.

Safael, N., Saidi-Mehrabad, M., Babakhani, M. (2007). Designing cellular manufacturing systems under dynamic and uncertain conditions, *Journal of Intelligent Manufacturing*, **18**, 383-399.

Sandhir, R. P., Kumar, S. (2010). Dynamic fuzzy c-means (dFCM) clustering for continuously varying data environments, Fuzzy Systems (FUZZ), IEEE International Conference, July 18-23.

Sarker, B. R. (2001). Measures of grouping efficiency in cellular manufacturing systems, *European Journal of Operation Research*, **130**, 588-611.

Sarker, B. R., Li, Z. (2001). Job routing and operations scheduling: a network-based virtual cell formation approach, *Journal of Operational Research Society*, **52**, 673-681.

Saxena, L. K., Jain, P. K. (2011). Dynamic cellular manufacturing systems design-a comprehensive model, *International Journal of Advanced Manufacturing Technology*, **53**, 11-34.

Selim, H. M., Askın, R. G., Vakharia, A. S. (1998). Cell formation in group technology: review, evaluation and directions for future research, *Computers and Industrial Engineering*, **34(1)**, 3-20.

Seifoddini, H. (1989). Single linkage versus average linkage clustering in machine cells formation applications, *Computers and Industrial Engineering*, **16(3)**, 419-426.

Seifoddini, H., Wolfe, P. M. (1986). Application of the similarity coefficient method in group technology, *IIE Transactions*, **18(3)**, 266-270.

Srinivasan, G., Narendran, T. T. (1991). GRAFICS-a nonhierarchical clustering algorithm for group technology, *International Journal of Production Research*, **29**, 463-478.

Srinivasan, G. (1994). A clustering algorithm for machine cell formation in group technology using minimum spanning trees, *International Journal of Production Research*, **32(9)**, 2149-2158.

Srinivasan, G., Narendran, T., Mahadevan, B. (1990). An assignment model for the part-families problem in group technology, *International Journal of Production Research*, **28**, 145-152.

Stanfel, L. (1985). Machine clustering for economic production, *Engineering Costs and Production Economics*, **9**, 73-78.

Suresh, N. C., Slomp, J. (2005). Performance comparison of virtual cellular manufacturing with functional and cellular layouts in DRC settings, *International Journal of Production Research*, **43(5)**, 945-979.

Tam, K. Y. (1990). An operation sequence based similarity coefficient for part families formation, *Journal of Manufacturing Systems*, **9**, 55-68.

Turker, U. (1993). Static and dynamic considerations of part family and machine cell formations, Masters' diss., University of Ottawa.

Vakharia, A. J., Moily, J. P., Huang, Y. (1999). Evaluating virtual cells and multistage flow shops: an analytical approach, *The International Journal of Flexible Manufacturing Systems*, **11**, 291-314.

Waghodekar, P. H., Sahu, S. (1984). Machine-component cell formation in group technology: MACE, *International Journal of Production Research*, **22**, 937-948.

Wang, J., Roze, C. (1997). Formation of machine cells and part families: a modified p-median model and comparative study, *International Journal of Production Research*, **35(5)**, 1259-1286.

Wellman, M. P. (1993). A Market-oriented programming environment and its application to distributed multicommodity flow problems, *Journal of Artificial Intelligence Research*, **1**, 1-23.

Wu, T., Chang, C., Chung, S. (2008). A simulated annealing algorithm for manufacturing cell formation problems, *Expert Systems with Applications*, **34**, 1609-1617.

Xiangyong, L., Baki, M. F., Aneja, Y. P. (2010). An ant colony optimization metaheuristic for machine-part cell formation problems, *Computers and Operation Research*, **37**, 2071-2081.

Yin, Y., Yasuda, K. (2006). Similarity coefficient methods applied to the cell formation problem: a taxonomy and review, *International Journal of Production Economics*, **101**, 329-352.

Younes, A. (2006). Adapting evolutionary approaches for optimization in dynamic environments, PhD diss., University of Waterloo.

APPENDIX A

In section 3 we have presented an example which illustrates the steps of agent based dynamic part family formation algorithm. Let’s use the solution of the illustrative example of section 3 to explain the machine allocation procedure. The machine-part incidence matrix considering obtained part families using the agent based dynamic part family formation algorithm is given in Figure A.1.

	Parts in Part Family 1							Parts in Part Family 2					Parts in Part Family 3					
	P2	P5	P7	P9	P10	P13	P14	P1	P3	P6	P8	P17	P4	P11	P12	P15	P16	P18
M1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0
M2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0
M3	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
M4	0	0	0	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0
M5	0	0	1	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0
M6	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	0	0
M7	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0	0
M8	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
M9	0	1	0	0	0	0	0	1	0	1	0	0	1	1	1	0	1	1
M10	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure A.1 Machine-part incidence matrix considering obtained part families

First of all we need to determine the sum of the voids and exceptional elements for each machine considering of the each part family. Figure A.2 shows these sums for each machine-part family pair.

	Part Family 1	Part Family 2	Part Family 3
	voids+exceptional elements	voids+exceptional elements	voids+exceptional elements
M1	**1+2=3**	4+7=11	5+7=12
M2	**0+1=1**	5+8=13	5+7=12
M3	**0+0=0**	5+7=12	6+7=13
M4	**4+2=6**	4+4=8	5+4=9
M5	5+5=10	**0+2=2**	6+7=13
M6	7+8=15	**0+3=3**	3+5=8
M7	7+5=12	**2+2=4**	4+3=7
M8	7+7=14	4+6=10	**0+1=1**
M9	6+7=13	3+6=9	**1+3=4**
M10	7+6=13	5+6=11	**0+0=0**

Figure A.2 Sum of voids and exceptional elements for each machine-part family pair

Machine 1, machine 2, machine 3, and machine 4 have the least sum of voids and exceptional elements if they are assigned to part family 1. Machine 5, machine 6, and machine 7 have the least sums if they are assigned to part family 2. Machine 8, machine 9, and machine 10 have the least sum of voids and exceptional elements if they are assigned to part family 3. Determined machines and part families for each cell are illustrated in Figure A.3.

	Cell 1							Cell 2					Cell 3					
	P2	P5	P7	P9	P10	P13	P14	P1	P3	P6	P8	P17	P4	P11	P12	P15	P16	P18
M1	**1**	**0**	**1**	**1**	**1**	**1**	**1**	1	0	0	0	0	0	0	0	0	1	0
M2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	1	0	0	0
M3	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0
M4	**0**	**0**	**0**	**1**	**1**	**0**	**1**	0	0	0	1	0	0	0	1	0	0	0
M5	0	0	1	1	0	0	0	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0
M6	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	**1**	0	1	1	1	0	0
M7	0	0	0	0	0	0	0	**0**	**1**	**1**	**0**	**1**	0	1	0	1	0	0
M8	0	0	0	0	0	0	0	0	0	1	0	0	**1**	**1**	**1**	**1**	**1**	**1**
M9	0	1	0	0	0	0	0	1	0	1	0	0	**1**	**1**	**1**	**0**	**1**	**1**
M10	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**

Figure A.3 Determined machines and part families for each cell

APPENDIX B

```
public class Part extends Agent
{

// Plain Variables
public int partNo;
public int nOfOperations;
public double dueDate;
public double[] disToPFMatrix;
public double aveDistToPF;
public double guessedProcTime;
public int lookforNPF;
public double partTime;
public double arrivalTime;
public int partType;
public int schNo;
public double totalTravellingTime;
public double fte;
public double dde;
public int processCompletedSc;
public double minSetupPart;
public int partFamilyBelonged;
public double aveDistToSPF;
public int quantity;
public int initial;
public int processCompletedCl;
public double totalOperationTime;
public int selectedMach;
public int lastMach;
```

// Collection Variables

```
public java.util.ArrayList <String > sequencesSt = new java.util.ArrayList<String>();
public java.util.ArrayList <Integer > sequences = new java.util.ArrayList<Integer>();
public java.util.ArrayList <Integer > sequencesForScheduling = new
java.util.ArrayList<Integer>();
public java.util.ArrayList <Integer > totalWorkOfP = new
java.util.ArrayList<Integer>();
public java.util.ArrayList < Integer > selectedMachines = new
java.util.ArrayList<Integer>();
```

// Dynamic (Flow/Auxiliary/Stock) Variables

```
public HyperArray MachMachDist = new HyperArray( Machines, Machines );
```

// Events

```
public EventTimeout _autoCreatedDS_xjal = new EventTimeout(this);
@Override
public String getNameOf( EventTimeout _e ) {
if ( _e == _autoCreatedDS_xjal ) return "Auto-created DataSets auto update event";
return super.getNameOf( _e );
}
@Override
public int getModeOf( EventTimeout _e ) {
if ( _e == _autoCreatedDS_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
return super.getModeOf( _e );
}
@Override
public double getFirstOccurrenceTime( EventTimeout _e ) {
if (
_e == _autoCreatedDS_xjal
) return getEngine().getStartTime();
return super.getFirstOccurrenceTime( _e );
}
@Override
public double evaluateTimeoutOf( EventTimeout _e ) {
```

```
if ( _e == _autoCreatedDS_xjal )  return 1;
return super.evaluateTimeoutOf( _e );
}
@Override
public void executeActionOf( EventTimeout _e ) {
if ( _e == _autoCreatedDS_xjal ) {
for (DataSet _ds : _ds_MachMachDist) {
_ds.update();
}
}
super.executeActionOf( _e );
}

// Statecharts
public Statechart stateOfPart = new Statechart( this, (short)2 );
@Override
public String getNameOf( Statechart _s ) {
if(_s == this.stateOfPart) return "stateOfPart";
return super.getNameOf( _s );
}
@Override
public void executeActionOf( Statechart _s ) {
if( _s == this.stateOfPart ) {
enterState( partInClustering, true );
return;
}
super.executeActionOf( _s );
}

// States of all statecharts
public static final short partInClustering = 0;
public static final short newPart = 1;
public static final short initialization = 2;
public static final short partInCluster = 3;
```

```
public static final short lookForNewPF = 4;
public static final short waitForDecision = 5;
public static final short determiningMachine = 6;
public static final short scheduling = 7;
public static final short scheduledPart = 8;
public static final short branch = 9;
@Override
public String getNameOfState( short _state ) {
switch( _state ) {
case partInClustering: return "partInClustering";
case newPart: return "newPart";
case initialization: return "initialization";
case partInCluster: return "partInCluster";
case lookForNewPF: return "lookForNewPF";
case waitForDecision: return "waitForDecision";
case determiningMachine: return "determiningMachine";
case scheduling: return "scheduling";
case scheduledPart: return "scheduledPart";
case branch: return "branch";
default: return super.getNameOfState( _state );
}
}
@Override
public boolean stateContainsState( short compstate, short simpstate ) {
if (compstate == partInClustering && (simpstate == newPart || simpstate ==
partInCluster || simpstate == waitForDecision || simpstate == lookForNewPF ||
simpstate == initialization)) {
return true;
}
return super.stateContainsState( compstate, simpstate );
}
@Override
public short getContainerStateOf( short _state ) {
switch( _state ) {
```

```
case newPart: return partInClustering;
case initialization: return partInClustering;
case partInCluster: return partInClustering;
case lookForNewPF: return partInClustering;
case waitForDecision: return partInClustering;
default: return super.getContainerStateOf( _state );
}
}
@Override
public void enterState( short _state, boolean _destination ) {
switch( _state ) {
case partInClustering: // (Composite state)
if ( _destination ) {
enterState( newPart, true );
}
return;
case newPart: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( newPart );
{
arrivalTime=time();
;}
transition5.start();
return;
case initialization: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( initialization );
{
initial=0;
for(int bn=0;bn<totalWorkOfP.size();bn++)
{
avaiMach.add(new ArrayList<Integer>());
}
for(int i=0;i<sequences.size();i++)
{
```

```
totalOperationTime=totalOperationTime+get_Main().aveProcTimeOfREs.get(seque
nces.get(i))*quantity;
}
guessedProcTime=fte*totalOperationTime;
dueDate=arrivalTime+dde*totalOperationTime;
if (get_Main().parts.nPartInClustering()==1)
{
get_Main().add_partFamilies();
get_Main().partFamilies.get((get_Main().partFamilies.size()-
1)).partInPartFamily.add(this);
get_Main().partFamilies.get((get_Main().partFamilies.size()-
1)).partFamilyNo=(get_Main().partFamilies.size()-1);
partFamilyBelonged=(get_Main().partFamilies.size()-1);
aveDisSPF();
aveDisPF();
}
else
{
aveDisPF();
get_Main().clusteringManager.partsInAuction.add(this);
send("hello",get_Main().clusteringManager);
}
;}
transition.start();
return;
case partInCluster: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( partInCluster );
{
get_Main().clusteringManager.onChange();
;}
transition1.start();
transition4.start();
return;
case lookForNewPF: // (Simple state (not composite))
```

```
stateOfPart.setActiveState_xjal( lookForNewPF );
{
aveDisSPF();
aveDisPF();
if(((aveDistToPF<aveDistToSPF)&&(aveDistToPF<=get_Main().partFamilies.get(p
artFamilyBelonged).threshold))||((get_Main().partFamilies.get(partFamilyBelonged).
threshold<aveDistToSPF)&&(get_Main().partFamilies.get(partFamilyBelonged).par
tInPartFamily.size()>1)))
{
get_Main().clusteringManager.partsInAuction.add(this);
}
;}
transition2.start();
return;
case waitForDecision: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( waitForDecision );
{
get_Main().clusteringManager.onChange();
;}
transition3.start();
return;
case determiningMachine: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( determiningMachine );
{
int totalAvaiMach=avaiMach.get(sequencesForScheduling.get(0)).size();
int
minQueue=get_Main().machines.get(avaiMach.get(sequencesForScheduling.get(0)).
get(0)).partsInQueue.size();
int
minQueMach=get_Main().machines.get(avaiMach.get(sequencesForScheduling.get(
0)).get(0)).machineNo;
for(int i=1;i<totalAvaiMach;i++)
{
```

```
if(get_Main().machines.get(avaiMach.get(sequencesForScheduling.get(0)).get(i)).par
tsInQueue.size()<minQueue)
{
minQueue=get_Main().machines.get(avaiMach.get(sequencesForScheduling.get(0)).
get(i)).partsInQueue.size();
minQueMach=get_Main().machines.get(avaiMach.get(sequencesForScheduling.get(
0)).get(i)).machineNo;
}
}
selectedMach=minQueMach;
selectedMachines.add(selectedMach);
if(lastMach>-1)
{
partTime=time()+MachMachDist.get((lastMach),(selectedMach));
totalTravellingTime=totalTravellingTime+MachMachDist.get((lastMach),(selected
Mach));
}
;}
transition7.start();
return;
case scheduling: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( scheduling );
{
get_Main().machines.get(selectedMach).partsInQueue.add(this);
if(get_Main().machines.get(selectedMach).partsInQueue.size()==0)
{
get_Main().machines.get(selectedMach).onChange();
}
;}
transition10.start();
return;
case scheduledPart: // (Simple state (not composite))
stateOfPart.setActiveState_xjal( scheduledPart );
{
```

```
get_Main().partFamilies.get(partFamilyBelonged).schPartNum=get_Main().partFami
lies.get(partFamilyBelonged).schPartNum+1;
get_Main().partFamilies.get(partFamilyBelonged).onChange();
partTime=time();
get_Main().numberOfSchPart=get_Main().numberOfSchPart+1;
schNo=get_Main().numberOfSchPart;
;}
return;
case branch: // (Branch)
if (
sequencesForScheduling.size()==0
 ) { // transition8
enterState( scheduledPart, true );
return;
}
// transition9 (default)
enterState( determiningMachine, true );
return;
default:
super.enterState( _state, _destination );
return;
}
}
@Override
public void exitState( short _state, Transition _t, boolean _source, Statechart
_statechart) {
switch( _state ) {
case partInClustering: // (Composite state)
if ( _source ) exitInnerStates(_state, _statechart);
return;
case newPart: // (Simple state (not composite))
if ( !_source || _t != transition5) transition5.cancel();
return;
case initialization: // (Simple state (not composite))
```

```
if ( !_source || _t != transition) transition.cancel();
return;
case partInCluster: // (Simple state (not composite))
if ( !_source || _t != transition1) transition1.cancel();
if ( !_source || _t != transition4) transition4.cancel();
{
processCompletedCl=0;
;}
return;
case lookForNewPF: // (Simple state (not composite))
if ( !_source || _t != transition2) transition2.cancel();
return;
case waitForDecision: // (Simple state (not composite))
if ( !_source || _t != transition3) transition3.cancel();
return;
case determiningMachine: // (Simple state (not composite))
if ( !_source || _t != transition7) transition7.cancel();
return;
case scheduling: // (Simple state (not composite))
if ( !_source || _t != transition10) transition10.cancel();
return;
case scheduledPart: // (Simple state (not composite))
return;
default:
super.exitState( _state, _t, _source, _statechart);
return;
}
}
public TransitionTimeout transition2 = new TransitionTimeout( this );
@Override
public String getNameOf( TransitionTimeout _t ) {
if ( _t == transition2 ) return "transition2";
return super.getNameOf( _t );
}
```

```
@Override
public Statechart getStatechartOf( TransitionTimeout _t ) {
if ( _t == transition2 ) return stateOfPart;
return super.getStatechartOf( _t );
}
@Override
public void executeActionOf( TransitionTimeout _t ) {
if ( _t == transition2 ) {
exitState( lookForNewPF, _t, true, stateOfPart );
enterState( waitForDecision, true );
return;
}
super.executeActionOf( _t );
}
@Override
public double evaluateTimeoutOf( TransitionTimeout _t ) {
if ( _t == transition2 ) return 0;
return super.evaluateTimeoutOf( _t );
}
public TransitionCondition transition5 = new TransitionCondition( this );
public TransitionCondition transition = new TransitionCondition( this );
public TransitionCondition transition1 = new TransitionCondition( this );
public TransitionCondition transition3 = new TransitionCondition( this );
public TransitionCondition transition4 = new TransitionCondition( this );
public TransitionCondition transition7 = new TransitionCondition( this );
public TransitionCondition transition10 = new TransitionCondition( this );
@Override
public String getNameOf( TransitionCondition _t ) {
if ( _t == transition5 ) return "transition5";
if ( _t == transition ) return "transition";
if ( _t == transition1 ) return "transition1";
if ( _t == transition3 ) return "transition3";
if ( _t == transition4 ) return "transition4";
if ( _t == transition7 ) return "transition7";
```

```
if ( _t == transition10 ) return "transition10";
return super.getNameOf( _t );
}
@Override
public Statechart getStatechartOf( TransitionCondition _t ) {
if ( _t == transition5 ) return stateOfPart;
if ( _t == transition ) return stateOfPart;
if ( _t == transition1 ) return stateOfPart;
if ( _t == transition3 ) return stateOfPart;
if ( _t == transition4 ) return stateOfPart;
if ( _t == transition7 ) return stateOfPart;
if ( _t == transition10 ) return stateOfPart;
return super.getStatechartOf( _t );
}
@Override
public boolean testGuardOf( TransitionCondition _t ) {
if ( _t == transition1 ) return
get_Main().clusteringManager.stateOfClusteringManager.isStateActive(ClusteringM
anager.waitingForApplication)&&(get_Main().partFamilies.get(partFamilyBelonged
).startScheduling==0);
return super.testGuardOf(  _t );
}
@Override
public void executeActionOf( TransitionCondition _t ) {
if ( _t == transition5 ) {
exitState( newPart, _t, true, stateOfPart );
enterState( initialization, true );
return;
}
if ( _t == transition ) {
exitState( initialization, _t, true, stateOfPart );
enterState( partInCluster, true );
return;
}
```

```
if ( _t == transition1 ) {
exitState( partInCluster, _t, true, stateOfPart );
enterState( lookForNewPF, true );
return;
}
if ( _t == transition3 ) {
exitState( waitForDecision, _t, true, stateOfPart );
enterState( partInCluster, true );
return;
}
if ( _t == transition4 ) {
exitState( partInCluster, _t, true, stateOfPart );
exitState( partInClustering, _t, false, stateOfPart );
enterState( determiningMachine, true );
return;
}
if ( _t == transition7 ) {
exitState( determiningMachine, _t, true, stateOfPart );
enterState( scheduling, true );
return;
}
if ( _t == transition10 ) {
exitState( scheduling, _t, true, stateOfPart );
{
lastMach=selectedMach;
sequencesForScheduling.remove(0);
processCompletedSc=0;
;}
enterState( branch, true );
return;
}
super.executeActionOf( _t );
}
@Override
```

```
public boolean testConditionOf( TransitionCondition _t ) {
if ( _t == transition5 ) return
(initial==1)&&(get_Main().parts.get(0).lookforNPF==0)&&(get_Main().parts.nIniti
alization()==0)&&(get_Main().clusteringManager.stateOfClusteringManager.isState
Active(ClusteringManager.waitingForApplication));
if ( _t == transition ) return
partFamilyBelonged>=0;
if ( _t == transition1 ) return lookforNPF==1;
if ( _t == transition3 ) return processCompletedCl==1;
if ( _t == transition4 ) return
get_Main().partFamilies.get(partFamilyBelonged).startSchedulingP==1;
if ( _t == transition7 ) return time()>=partTime;
if ( _t == transition10 ) return processCompletedSc==1;
return super.testConditionOf( _t );
}
```

// Functions

```
void aveDisPF( ) {
disToPFMatrix=new double[get_Main().partFamilies.size()];
int pFFound;
for (int y=0;y<get_Main().partFamilies.size();y++)
{
if (y==partFamilyBelonged)
{
disToPFMatrix[y]=10000000.0;
}
else
if(get_Main().partFamilies.get(y).stateOfPartFamily.isStateActive(PartFamily.partFa
milyInScheduling))
{
disToPFMatrix[y]=10000000.0;
}
else
{
```

```
double avedis=0.0;
double disToPF=0.0;
int sizeOfPF=0;
for (int z=0;z<get_Main().partFamilies.get(y).partInPartFamily.size();z++)
{
int n=nOfOperations;
int m=get_Main().partFamilies.get(y).partInPartFamily.get(z).nOfOperations;
int [][] dis1Matrix=new int[n+1][m+1];
dis1Matrix[0][0]=0;
for (int a=1;a<=m;a++)
{
dis1Matrix[0][a]=a;
}
for (int b=1;b<=n;b++)
{
dis1Matrix[b][0]=b;
}
for (int k=1;k<=n;k++)
{
for (int j=1;j<=m;j++)
{
int substitute=0;
int delete=0;
int addition=0;
if (sequencesSt.get(k-
1).equals(get_Main().partFamilies.get(y).partInPartFamily.get(z).sequencesSt.get(j-
1)))
{
substitute=dis1Matrix[k-1][j-1];
}
else
{
substitute=dis1Matrix[k-1][j-1]+1;
}
```

```
delete=dis1Matrix[k-1][j]+1;
addition=dis1Matrix[k][j-1]+1;
dis1Matrix[k][j]=min(substitute, min(delete,addition));
}
}
double dis1=dis1Matrix[n][m];
int same=0;
double intersection=0;
double union=0;
for (int p=0;p<n;p++)
{
for (int r=0;r<m;r++)
{
if
(sequencesSt.get(p).equals(get_Main().partFamilies.get(y).partInPartFamily.get(z).se
quencesSt.get(r)))
{
same=same+1;
break;
}
}
}
intersection=same;
union=n+m-intersection;
double dis2=1-(intersection/union);
double dissimilarity=0.5*dis1+0.5*dis2;
disToPF=disToPF+dissimilarity;
sizeOfPF=sizeOfPF+1;
if (get_Main().partFamilies.get(y).partInPartFamily.size()==sizeOfPF)
{
avedis=disToPF/sizeOfPF;
}
}
disToPFMatrix[y]=avedis;
```

```
}
}
double aa=disToPFMatrix[0];
int mindis=0;
for (int bb=0;bb<get_Main().partFamilies.size();bb++)
{
if (aa>disToPFMatrix[bb])
{
aa=disToPFMatrix[bb];
mindis=bb;
}
}
aveDistToPF=aa;
pFFound=mindis;
}
void aveDisSPF(  ) {
if (get_Main().partFamilies.get(partFamilyBelonged).partInPartFamily.size()==1)
{
aveDistToSPF=10000000.0;
}
else
{
double avedis=0.0;
double disToPF=0.0;
int sizeOfPF=0;
for (int
z=0;z<get_Main().partFamilies.get(partFamilyBelonged).partInPartFamily.size();z+
+)
{
int n=nOfOperations;
int
m=get_Main().partFamilies.get(partFamilyBelonged).partInPartFamily.get(z).nOfOp
erations;
int [][] dis1Matrix=new int[n+1][m+1];
```

```
dis1Matrix[0][0]=0;
for (int a=1;a<=m;a++)
{
dis1Matrix[0][a]=a;
}
for (int b=1;b<=n;b++)
{
dis1Matrix[b][0]=b;
}
for (int k=1;k<=n;k++)
{
for (int j=1;j<=m;j++)
{
int substitute=0;
int delete=0;
int addition=0;
if (sequencesSt.get(k-
1).equals(get_Main().partFamilies.get(partFamilyBelonged).partInPartFamily.get(z).
sequencesSt.get(j-1)))
{
substitute=dis1Matrix[k-1][j-1];
}
else
{
substitute=dis1Matrix[k-1][j-1]+1;
}
delete=dis1Matrix[k-1][j]+1;
addition=dis1Matrix[k][j-1]+1;
dis1Matrix[k][j]=min(substitute, min(delete,addition));
}
}
double dis1=dis1Matrix[n][m];
int same=0;
double intersection=0;
```

```
double union=0;
for (int p=0;p<n;p++)
{
for (int r=0;r<m;r++)
{
if
(sequencesSt.get(p).equals(get_Main().partFamilies.get(partFamilyBelonged).partIn
PartFamily.get(z).sequencesSt.get(r)))
{
same=same+1;
break;
}
}
}
intersection=same;
union=n+m-intersection;
double dis2=1-(intersection/union);
double dissimilarity=0.5*dis1+0.5*dis2;
disToPF=disToPF+dissimilarity;
sizeOfPF=sizeOfPF+1;
if
(get_Main().partFamilies.get(partFamilyBelonged).partInPartFamily.size()==sizeOf
PF)
{
avedis=disToPF/(sizeOfPF-1);
}
}
aveDistToSPF=avedis;
}
}
```

PERSONAL INFORMATION

Name and Surname: Latife Görkemli
Natioality: Turkish (TC)
Birth place and date: Kayseri, 1985
Marial status: Single
Phone number: +90 537 468 06 54
Email: lgorkemli@erciyes.edu.tr

EDUCATION

	Graduate school	Year
Master	Erciyes University (Ind. Eng.)	2009
Bachelor	Erciyes University (Ind. Eng.)	2007
High School	N.M. Küçükçalık Anadolu Lisesi	2003

Work experience

	Place	Enrollment
2003-Present	Erciyes University	Research Assistant

PUBLICATIONS

International Journals

Baykasoğlu, A. Gorkemli, L., 2014. Dynamic virtual cellular manufacturing through agent based modelling, Submitted to International Journal of Production Research.

Baykasoğlu, A. Gorkemli, L., 2013. Agent based dynamic part family formation for cellular manufacturing applications, Submitted to International Journal of Production

Research.

Baykasoglu, A., Ozbakir, L., Gorkemli, L., Gorkemli, B., 2012. Multi-colony ant algorithm for parallel assembly line balancing with fuzzy parameters, Journal of Intelligent and Fuzzy Systems, 23(6), 283-295.

Özbakır L., Baykasoğlu A., Görkemli B., Görkemli L., 2011. Multiple-colony ant algorithm for parallel assembly line balancing problem, Applied Soft Computing, 11(3), 3186-3198.

Görkemli, L., Kapan Ulusoy, S., 2010. Fuzzy Bayesian reliability and availability analysis of production systems, Computers & Industrial Engineering, 59, 4, 690-696.

Internatinoal Conferences

Baykasoglu, A., Gorkemli, L., Formation of dynamic virtual manufacturing cells through agent based modeling, YA/EM 2013: Yöneylem Arastirmasi / Endüstri Mühendisligi Kongresi 33. Ulusal Kongresi, International IIE Conference, Grand Cevahir Otel ve Kongre Merkezi, 26-28 Haziran 2013, Istanbul, (ISBN: 978-605-61427-8-9), 2013. (abstract)

Baykasoglu, A., Ozbakir, L., Gorkemli, L., Gorkemli, B., Multiple ant colony algorithm for balancing parallel assembly lines with fuzzy parameters, FUZZYSS'11: 2nd International Fuzzy Systems Symposium, (ed., Gokceoglu C., Aladag HC., Akgun A) Hacettepe University, Cultural Center, November 17-18, 2011, Ankara, Turkey, pp. 163-168.

Baykasoglu, A., Durmusoglu, Z.D.U., Gorkemli, L., Solving vehicle deployment planning problem by using agent based simulation modeling, 2nd International Symposium on Computing in Science & Engineering, June, 1-4, 2011, Gediz University Publications, editor: M. Gunes, ISBN:978-605-61394-2-0, pp.338-340, Kusadasi, Aydin, Turkey.

Baykasoğlu, A., Özbakır, L., Görkemli, L., Görkemli, B., 2009, Balancing parallel assembly lines via ant colony optimization, 39th International Conference on Computers & Industrial Engineering (CIE39), 6-9 July 2009, Troyes, France, p.p.

512-517.

Görkemli, L., Ulusoy, S., Bayesian analysis of manufacturing process reliability, Innovative Approaches to University&Small and Medium Enterprises (SMES) Cooperation, Workshop I, 19-20 Mart, Litvanya, 2009. (abstract)

Ulusoy, S., Görkemli, L., 2008. Analysis of maintenance data of an airline. HDM 2008: International Conference on Multivariate Statistical Modeling and High Dimensional Data Mining, 19-23 June 2008, Erciyes University, Kayseri. (abstract)

National Conferences

Baykasoglu, A., Durmusoglu, Z. D. U., Gorkemli L., Dinamik araç yerleştirme problemleri için etmen tabanlı çözüm stratejileri geliştirilmesi, 13. Üretim Araştırmaları Sempozyumu, Bildiriler Kitabı, 25-27 Eylül 2013, Sakarya Universitesi Esentepe Kampüsü Kongre ve Kültür Merkezi, pp. 269-278.

Baykasoglu, A., Durmusoglu, Z.D.U., Gorkemli, L., Etmen tabanlı benzetim: ANYLOGIC^{TM} yazılımı ve örnek bir uygulama, Endüstri Mühendisliği Yazılımları ve Uygulamaları Kongresi, İzmir, 30 Eylül-01/02 Ekim 2011, TMMOB Makina Mühendisleri Odası Yayın No: E/2011/559, pp. 197-204.

Görkemli, L., Görkemli, B., 2010. Montaj hatlarının hazırlık süreleri dikkate alınarak karınca kolonisi algoritması ile dengelenmesi ve çizelgelenmesi. YA/EM 2010 Yöneylem Araştırması/Endüstri Mühendisliği 30. Ulusal Kongresi, İstanbul. (bildiri özeti)

Özbakır, L., Baykasoğlu, A., Görkemli, B., Görkemli, L., 2009. Çok kriterli paralel montaj hattı dengeleme problemine çok kolonili karınca optimizasyonu yaklaşımı. YA/EM 2009 Yöneylem Araştırması/Endüstri Mühendisliği 29. Ulusal Kongresi, Ankara. (tam metin)

Ulusoy, S., Görkemli, L., 2009. Bayes ağları ile uçuş gecikmelerinin analizi. YA/EM 2009 Yöneylem Araştırması/Endüstri Mühendisliği 29. Ulusal Kongresi, Ankara. (bildiri özeti)

Baykasoğlu, A., Özbakır, L., Görkemli, L., Görkemli, B., 2008. Parallel montaj hattı dengelemede çok kolonili karınca kolonisi optimizasyonu. YA/EM 2008 Yöneylem Araştırması/Endüstri Mühendisliği 28. Ulusal Kongresi, İstanbul. (tam metin)

Göleç, A., Görkemli, L., 2007. Panel mobilya üretiminde stratejik rekabet gücünü etkileyen ergonomik konular. 13. Ulusal Ergonomi Kongresi, Kayseri. (tam metin)

Görkemli, L., Kara, G., Göleç, A., 2007. İstikbal Mobilya A.Ş.'de yatak montaj hattı tasarımı. YA/EM'2007 Yöneylem Araştırması ve Endüstri Mühendisliği 27. Ulusal Kongresi, İzmir. (tam metin)

FOREIGN LANGUAGE

English
German

HOBBIES

Jogging and playing table tennis.