AN INFORMATION THEORY BASED CAPACITY CALCULATION METRIC FOR MULTI AGENT AND PETRI NET MODELED SYSTEMS

A MASTER'S THESIS

in

Software Engineering

Atılım University

by

DOĞUŞ BEBEK

NOVEMBER 2009

AN

INFORMATION THEORY BASED CAPACITY CALCULATION METRIC

FOR

MULTI AGENT AND PETRI NET MODELED SYSTEMS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

DOĞUŞ BEBEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

NOVEMBER 2009

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

Prof. Dr. İbrahim AKMAN

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ali Yazıcı

Head of Department

This is to certify that we have read the thesis "An Information Theory Based Capacity Calculation Metric for Multi Agent and Petri Net Modeled Systems" submitted by "Doğuş Bebek" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Hürevren KILIÇ

Supervisor

Examining Committee Members

Asst. Prof. Dr. Hürevren KILIÇ	______________________
Asst. Prof. Dr. Çiğdem Turhan	______________________
Asst. Prof. Dr. Murat Koyuncu	______________________
Asst. Prof. Dr. Sanjay Misra	______________________
Assoc. Prof. Dr. Mohammed Rehan	______________________

Date: 16.11.2009

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Doğuş BEBEK

Signature:

ABSTRACT

AN

INFORMATION THEORY BASED CAPACITY CALCULATION METRIC

FOR MULTI AGENT AND PETRI NET MODELED SYSTEMS

Bebek, Doğuş

M.S., Software Engineering Department

Supervisor: Asst.Prof.Dr. Hürevren KILIÇ

November 2009, 53 pages

This thesis proposes a metric based on Discrete Noiseless Channel concept of Information Theory to calculate the information capacity of Multiagent Systems and Petri Net modeled systems. Proposed metric is a design time metric and it calculates the maximum runtime information capacity of given system. Also, proposed metric with inefficiency and inactivity values, can be used to compare different designs or can be used to evaluate the possible modifications planning to be made on a system. Multiagent systems concept is a new paradigm in the software engineering and communication of agents in a multiagent system is a popular research area. Application of the proposed metric to the multiagent systems domain explained by first using the agent communication protocols and then using the multiagent communication topologies. Petri Nets are generic design tools for designing discrete event systems and in the literature there are lots of works done about designing software systems as Petri Nets. Application of proposed metric to Petri Net domain explained by using a Petri Net designed cruise control system example. With the application of proposed metric to the Petri Net domain, it is aimed to generalize the application area of the metric.

Keywords: Information Theory, Multi Agent Systems, Software Engineering, Petri Nets, Software Metrics, Information Capacity Calculation.

ÖZ

ÇOK ERKİNLİ VE PETRİ AĞLARI İLE MODELLENMİŞ SİSTEMLER İÇİN BİLGİ TEORİSİ TABANLI BİR KAPASİTE HESAPLAMA METRİĞİ

Bebek, Doğuş

Yüksek Lisans, Yazılım Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Hürevren KILIÇ

Kasım 2009, 53 sayfa

Bu tez çalışması, Bilgi Teorisinin Ayrık Parazitsiz Kanak konseptini baz alarak çok erkinli ve Petri Ağları ile modellenmiş sistemler için bir kapasite hesaplama metriği önermektedir. Önerilen metrik, ilgili sistemin çalışma anındaki maksimum bilgi kapasitesini sistemin dizayn aşamasında hesaplamaktadır. Ayrıca, ilgili metriğin verimsizlik ve durgunluk değerleri kullanılarak farklı tasarımlar karşılaştırılabilir ya da ilgili sistem üzerinde yapılması planlanan değişiklikler değerlendirilebilir. Çok erkinli sistemler paradigması yazılım mühendisliği alanında yeni bir paradigmadır ve çok erkinli sistemler içindeki erklerin iletişimi popular bir araştırma sahasıdır. Önerilen metriğin çok erkinli sistemlere uygulanması öncelikli olarak erkinler arası iletişim protokolleri daha sonrada çok erkinli iletişim topolojileri kullanılarak yapılmıştır. Peti Ağları, ayrık olay sistemlerinin modellenmesi için kullanılan araçlardır ayrıca literatürde yazılım sistemlerinin Petri Ağları ile modellenmesi üzerine bir çok çalışma mevcutdur. Önerilen metriğin, Petri ağları ile modellenmiş sistemlere uygulanması seyir kontrol sistemi örneği kullanılarak anlatılmıştır. Önerilen metriğin Petri Ağlarına uygulanması ile metriğin kullanım alanının genelleştirilmesi amaçlanmıştır.

Anahtar Kelimeler: Bilgi Teknolojisi, Çok Erkinli Sistem, Yazılım Mühendisliği, Yazılım Metrikleri, Petri Ağları, Metrik, Bilgi Kapasitesi Hesaplaması

To My Parents

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Asst.Prof.Dr. Hürevren KILIÇ for his guidance, patience an belief in this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Information flow based software systems are systems where run time behavior of the system is defined by the messaging between the elements of the system. By saying elements we mean, components if that is a component based software or for example, agents if it is a multi – agent system, etc. Similarly, object – oriented software systems can also be classified as information flow based systems since there is always flow of data between the classes, which defines the run time behavior of the system.

In 1948, Shannon proposed the discrete noiseless channel capacity metric as part of his information theory, to measure the information flow capacity of a network which allows transition of information through the network without any error [4]. Shannon's proposed metric, measures the maximum amount of information that can be transmitted over a channel in a unit of time (unit cost). The unit of the measurement is defined as nats in Shannon's original work and this unit can also be interpreted as bit(s)/sec. One application of channel capacity metric of Shannon to the specific area of software engineering called, Component Based Software Design, is done by Şeker and Tanık [1]. In their work, they have applied noiseless channel capacity metric to the software systems designed by using COTS (Commercial of the Shelf) products, in order to "provide reliable, practical, and cost – effective assessment method to reduce the increasing cost of quality assessment for software systems". They measured inefficiency and inactivity values of the system using capacity metric to give the system integrator an idea about the impacts of modifying or changing the components of the system. By taking the current status of the system (i.e. inefficiency and inactivity values) as base, they evaluate how changes or modifications on the components of the system affect the inefficiency and inactivity values of the system. Although their study was limited with the component based software systems, as they stated in their work it is not a must to limit the application of information theory to the software engineering world with the component based systems [1]. With the inspiration of their work [1] and knowledge of the author on the multiagent systems, it is first decided to apply proposed metric to the multiagent systems domain. Multiagent systems concept is a new paradigm in the software engineering for creating software in distributed and open environments. By applying this proposed metric, it is aimed to provide system designers opportunity of evaluating their design amongst the other equivalent designs with the inefficiency and inactivity metrics.

A Multi agent system is defined as *"a loosely coupled network of problem solvers that interact to solve the problems that are beyond the individual capabilities"* by [2]. To realize their individual goals, in the multi agent environments, agents communicate with each other and this communication is done by using some predefined communication protocols. In other words, these protocols define the rules of the communication and at the end of this communication, for instance if it is a cooperative environment, agents share their tasks to achieve their goals. Therefore it is decided to apply Shannon's metric to the Multiagent Systems domain to quantify the performance of the preferred agent communication protocol among its alternatives.

In this study, our starting point was the communication between two agents. But in real, multi agent environments include more than two agents. Communication between the agents in the environment is dependent to the topology of the environment. In other words, agents can communicate with each other if and only if, there are links between them, which is defined by the topology itself. From that point, we can apply the proposed metric to the multi agent environment topologies to quantify their performance between them, which is the next step of this work.

Second part of this work is about to apply proposed metric to the Petri Net modeled systems. “Petri Nets, as graphical mathematical tools, provides uniform environment for modeling, formal analysis and design of discrete event systems”[3]. Any system, which can be described as flow of events, can be modeled using Petri Nets. There are lots of works concentrated on designing software systems by using Petri Nets. Besides the software engineering, since Petri Nets are generalized design tools for discrete event systems, application of proposed metric to Petri Nets is going to provide the designers a new evaluation perspective especially amongst the alternative equivalent designs. In this thesis, application of proposed metric to Petri Net domain is going to be concentrated on the Petri Net modeled software systems.

In this work value or practical importance of proposed metric has been evaluated from the designer’s perspective. Designer is going to have the opportunity of evaluate his/her design against other alternatives with inactivity and inefficiency values of the metric. Also with these inefficiency and inactivity values, when any modification or change emerged, designer can foresee how system will be affected by means of cost effectiveness and performance which is very important because system designer is going to have an idea about the runtime capacity of the system during the design time.

Information capacity value of the evaluated system does not give the designer conclusive idea about the system’s performance. But with the use of proposed metric, a common sense about the capacity concept is going to be developed. On the other hand, inefficiency and inactivity values can be used compare two system in the design time which is important from the designer’s perspective.

Application of Shannon’s Information Theory to the software engineering area is mostly concentrated on measuring the complexity of the software systems based on the entropy concept.

Abdelwahab Hamou-Lhadj [21] in his work, "Measuring the Complexity of Traces Using Shannon Entropy" introduced new metrics based on the concept of entropy to discuss how complexity of a trace can be reduced based on presented metrics.

Ned Chapin [22] in his work, "Entropy–Metric For Systems With COTS Software" measures the complexity of the interaction of components of the system. In this work L-Metric is offered to be used as an analysis tool through the testing phases in both the system development and maintenance life-cycles.

Introduction of Combinatorial Capacity Calculation of Discrete Noiseless Channel and the related studies are as follows and due to the references below one can conclude that this study is the first application of Combinatorial Capacity Calculation metric to Multiagent systems and Petri Net modeled systems.

Khandekar.A, McEliece.R, Rodemich.E [5], discussed the Shannon’s Capacity Calculation Metric. Their work is largely a tutorial to Shannon’s Information Theory and also includes new ideas such as “Partition Function Technique” which is used in this thesis.

C.Pimentel and Bartolomeu [23] , In their work, "A Combinatorial Approach to Finding the Capacity of the Discrete Noiseless Channel”, they have demonstrated the application of combinatorial techniques to finding the Shannon capacity of the discrete noiseless channel. In our derivations, the case of noninteger symbol durations, as introduced by Csiszár, is considered.

Application of Combinatorial Capacity Calculation metric for Discrete Noiseless Channel concept is used as a software measure by R.Seker and M. Tanik and H.Kılıç and K. Öztoprak.

R.Seker and M. Tanik in their work [1], "An Information-Theoretical Framework for Modeling Component-Based Systems", brings modeling techniques from information theory, into component-based software (CBS) engineering. Information-theoretic representation and analysis techniques in general, noiseless information channel concepts in particular, are good candidates to be adopted to model the dynamic behavior of software components and quantify the interaction between them. This modeling approach is realized by first modeling the component integration units of CBS with cubic control flowgraphs. Then, a set of metrics, Shannon metrics, is defined.

H.Kılıç and K.Öztoprak, proposed a metric in their work for P2P networks based on Shannon's L-Channel capacity calculation idea. The metric calculates, the maximum rate of information that can be transmitted over the P2P network caused by protocol or overlay level connectivity [20].

The problem statement is, does Shannon's metric can be applied to the Multi Agent Systems and Petri Net modeled systems and what would be the practical actions that system designer could take by calculating capacity values of designed system. In this thesis first application of the Shannon's metric to the both Multi Agent and Petri Net modeled systems are going to be investigated and then the value of the application of this metric to these field is going to be discussed.

Rest of the thesis is organized as, in Chapter 2; background information about the Shannon's Information Theory is given. Application of the Shannon's metric to the agent communication protocols and Multi Agent Environment Topologies discussed in Chapter 3. In Chapter 4, you can find application of the Shannon's metric to the Petri Net modeled systems and in Chapter 5, you can find the conclusion part.

CHAPTER II

BACKGROUND INFORMATION

2.1 Shannon's Information Theory

Before starting to explain Shannon's discrete noiseless channel capacity metric, it is better to explain what are noise, channel, and discrete noiseless channel. Noise can be defined as the interference that does not allow to flow of information from one point to another. A channel is the place where information flows in it. So, noiseless channel is the channel which allows the flow of information without any error or loss and discrete noiseless channel is the channel that allows the noiseless transmission of finite sequence of information.

Information theory is developed by *C.E. Shannon*. One important part of information theory is discrete noiseless channel concept. In Shannon's original work discrete noiseless channel defined as a channel that allows the noiseless transmission of sequence of symbols chosen from a finite alphabet [4]. Each symbol in this alphabet has duration in time, possibly different for different symbols. The main question in here is the "What is the capacity of this channel?" We can define the capacity (in Information Theory context) as the maximum amount of information that can be transmitted over the discrete noiseless channel. Shannon's combinatorial capacity metric sets an upper bound to the information transmission rate of the channel. In Shannon's original work, unit of this capacity is the bit(s)/sec since the cost (due to symbol transmissions) measured unit in time. Therefore cost concept can be generalized and changed according to the calculation semantic of the area where this metric applied. From here one can easily say that, for any system that can be represented as Shannon Language (defined in following section) we can measure its runtime capacity just having the design of the system in hand.

In the following section there are series of mathematical definitions that explains the calculation of the capacity of the channel.

2.2 Combinatorial Capacity of a Discrete Noiseless Channel

The capacity of discrete noiseless channels is explained by following definitions and theorem. The following series of background definitions and theorem are compiled from [5]. For the original proof of Theorem 1 and background information see [4].

Definition 1: A discrete noiseless channel is a channel which allows the noiseless transmission of a sequence of symbols chosen from a finite alphabet A (called q-letter alphabet), each symbol having a certain duration in time, possibly different for different symbols.

Definition 2: A word of length *k* over *A* is a finite sting of *k* letters from *A*. If $\alpha = a_1 a_2 \ldots\ldots a_k$ is a such word, its duration is defined to be

$$\tau(\alpha) = \tau(a_1) + \tau(a_2) + \ldots. + \tau(a_k)$$

Definition 3: Shannon language is defined by a directed graph whose edges are labeled with letters from the alphabet A. The corresponding language L is then defined to be the set of words that result by reading off the edge labels on paths of the graph.

Definition 4: A language L is a collection of words over alphabet A. The discrete noiseless channel associated with L, the L-channel for short, is the channel which is only allowed to transmit sequences from L, where it transmits them without error.

Definition 5: Let *L* be a Shannon Language, the combinatorial capacity of the *L-Channel* is defined as

$$C_{comb} = \lim_{t\to\infty} \sup \frac{1}{t} \log(N(t))$$

where *N(t)* is the total number of words in *L* of duration *t.*

In Shannon's original work [4], an algebraic method of computing C_{comb} has been given. In [5], on the other hand, a more detailed and simpler (at least for the author of this thesis) calculation of C_{comb} has also been elaborated. Therefore, we prefer to continue with definitions from [5] which are finalized by Theorem 1 defining the C_{comb} calculation.

Definition 6: A directed graph (or digraph) G has vertex set V = $\{v_1, \ldots, v_m\}$ and branch set B = $\{b_1, \ldots, b_n\}$. Each branch $b \in B$ has an initial vertex init(b)$\in$ V and a final vertex fin(b)$\in$ V. The set of branches with initial vertex v and final vertex w is denoted by:

$$B_{v,w} = \{b \quad B: \text{init}(b) = v, \text{fin}(b) = w\}$$

Definition 7: A *path P* of length *k* from *v* to *w* in *G* is a sequence of *k* branches P=$b_1 b_2 \ldots b_k$ with init(b_1) = *v*, fin(b_1) = init(b_2), . . ., fin(b_{k-1}) = init(b_k), fin(b_k) = *w*. We write len(P) = *k*, init(P) = *v*, and fin(P) = *w*. Then, the set of paths of length *k* from *v* to w is denoted by:

$$B^{k}_{v,w} = \{P: \text{len}(P) = k, \text{init}(P) = v, \text{fin}(P) = w\}$$

Using a directed graph *G* and an alphabet *A*, we can generate a language by labeling each branch of *G* with an element of *A*.

Definition 8: Let $\lambda : B \to A$ be a labeling. λ is *right-resolving* iff for each vertex v, the labels on all branches with init(b) = *v* are distinct.

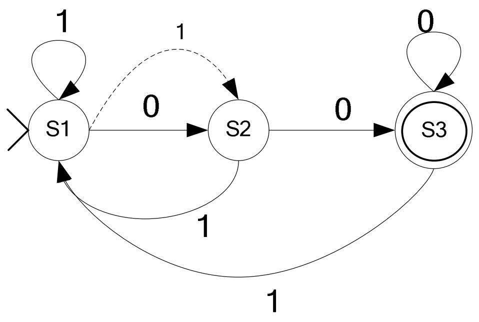


Figure 1 – A Right Resolving Directed Graph

Right resolving property is important, because if it is not satisfied then design will be non-deterministic. In Figure 1, above, you can see the directed graph having right resolving property. As explained in definition 8, for each vertex S1, S2 and S3 labels on the init(b)=v are distinct. On the other hand if we add the branch with dashed arrow to the graph, with input of 1, system could either stay in S1 or go to state S2 therefore during the run time we can not guarantee the global state of the system which conflicts with right resolving property.

Definition 9: The set of all possible path labels is called the *Shannon language* generated by G, and denoted by $L_{G,\lambda}$.

Definition 10: Let s be nonnegative real number, and for a given pair of vertices (v,w), *branch duration partition* function is defined as:

$$P_{v,w}(s) = \sum_{b \in B_{v,w}} e^{-s\tau(b)}$$

The functions $P_{v,w}(s)$ can be thought of as entries in a $M \times M$ matrix $P(s)$ where M is the number of verticies. The *spectral radius* (the magnitude of the largest eigenvalue) of the matrix P(s) is represented by $\rho(s)$ and it is also called the *partition function* for the language $L_{G,\lambda}$.

Theorem 1: The combinatorial capacity of the language is given by

$$C_{comb} = ln(s_0)$$

where s_0 is the unique solution to the equation $\rho(s) = 1$.

Alternatively, C_{comb} is the greatest positive solution of the equation (in nats),

$$q(s)=\det(I - P(s))=0$$

where I is the identity matrix and P(s) is the branch duration partition function.

Definition 11: The inefficiency (symbols generated per bit) of a Shannon language $L_{G,\lambda}$ is defined to be,

$$\zeta = 1/ C_{comb}$$

and is referred to as the inefficiency metric.

Definition 12: The inactivity of the Shannon language is defined to be

$$\varphi = 1 / 2^{Ccomb}$$

and is referred to as the inactivity metric.

<u>Example:</u> Below example is given to explain Shannon's combinatorial capacity, inefficiency and inactivity metrics in more detail and taken from [5]. In Figure 2, there is a directed graph G with vertices V= {v1, v2}, and branches B= {b1, b2, b3, b4, b5, b6}.

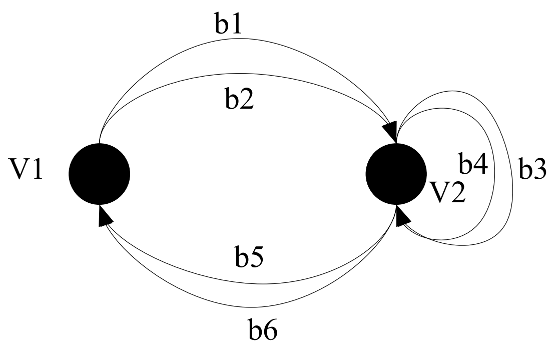


Figure 2 – Directed Graph G

Each branch is labeled by letters and each letter has certain duration in time. In Table1, you can see the branches, their starting and ending vertices, matching labels and channel durations of the labels.

From Definition 3, directed graph G in Figure 2 represents a Shannon Language, over an alphabet, say A, with letters A = {d, D, s, S}and each letter has duration in time as shown in Table1. Also from definition 8, we know that λ(G), labeling function, is right resolving. It is seen on Table 1; all branches having the same initial vertex have different labels.

Table 1 – Description of Directed Graph G

b	***init(v)***	***fin(v)***	***λ(b)***	***τ (b)***
b1	v1	v2	d	2
b2	v1	v2	D	4
b3	v2	v2	D	4
b4	v2	v2	d	2
b5	v2	v1	s	3
b6	v2	v1	S	6

From Definition 10, $M \times M$ matrix is 2x2 matrix since we have two vertices. Entries of this matrix define the connectivity of the graph. (0,0) entry of the matrix is 0 since there is no incoming arc from v1 to v1. (0,1) entry represents the arcs where init(v)=v1 and fin(v)=v2 and from the definition 10 is $e^{-2s} + e^{-4s}$. (1,0)th entry of the matrix is the arc where init(v)=v2 and fin(v)=v1 which is $e^{-3s} + e^{-6s}$. (1,1)th entry of the matrix is the arc where init(v)=v2 and fin(v)=v1 which is $e^{-2s} + e^{-4s}$. Finally branch duration function is as follows:

$$P_{v,w}(s) = \begin{bmatrix} 0 & e^{-2s} + e^{-4s} \\ e^{-3s} + e^{-6s} & e^{-2s} + e^{-4s} \end{bmatrix}$$

and from Theorem 1, combinatorial capacity of Shannon language L represented in Figure 2 is the maximum of the positive real solutions of the equation

$$q(s) = \det (I_2 - P(s)) = 0$$

where $I_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is the 2x2 identity matrix, and

$$q(s) = \begin{vmatrix} 0 & 1 - e^{-2s} - e^{-4s} \\ 1 - e^{-3s} - e^{-6s} & -e^{-2s} - e^{-4s} \end{vmatrix} = 0$$

is the determinant of I_2 - P(s), and

$$q(s) = 1 - e^{-2s} - e^{-4s} - e^{-6s} - e^{-8s} - e^{-12s} - e^{-24s} = 0$$

Numerical solution of q(s)=0, has been done by using following MATLAB commands.

```
syms s //define symbollic variable s
//Assignment of branch partition duration function
part_func = [0 exp(-2*s)+exp(-4*s); exp(-2*s)+exp(-4*s)
            exp(-  2*s)+exp(-4*s)]
// Determinant of I2-P(s)
determinant = det(eye(2)-part_func)
// calculates the reel roots of q(s)=0
result = double(solve(determinant))
```

The only positive real solution of q(s) is C_{comb} = 0.3736 nats. A Nat is a logarithmic unit of information or entropy and one Nat corresponds to about 1.44 bits. Therefore, information capacity C_{comb} = 0.5389 bits/sec. This number is the maximum amount of information (in bits) that can be found on the channel at any time stamp. In other words, for ant time instance t, the information flow amount on this channel can not exceed 0.5389 bits. That is the upper information flow bound for the channel in Figure 2.

CHAPTER III

APPLICATION OF PROPOSAL TO MULTIAGENT SYSTEMS

A Multi agent system is defined as *"a loosely coupled network of problem solvers that interact to solve the problems that are beyond their individual capabilities"*. To solve problems, agents come together in an environment and interact with each other. This interaction between the agents happens by the help of the agent communication.

Agent communication is a major research field of Artificial Intelligence (AI) and Multiagent systems. To formalize the agent communication, AI researchers by inspiring from the speech-act theory developed the agent communication languages. Speech-act theory is a theory of how utterances are used to achieve intensions and speech-act theorists introduced the illocutionary act concept which is a speech-act of doing something. It consists of verbs called performatives such as request, inform, promise etc. The most famous agent communication language is the one developed by FIPA (Foundation of Intelligent Physical Agents) [6] and called Agent Communication Language [7] (ACL in short).

ACL of FIPA is the latest speech-act based agent communication language and its predecessors are KQML and KIF. In this thesis, as an agent communication language FIPA's ACL is going to be used. ACL language has nearly twenty performatives to establish the communication between agents. All these performatives are derived from the two fundamental performative ***inform*** and ***request***. Details of agent communication language and speech-act theory are out of the scope of this thesis and one can find detailed information from [7].

Although agents have a common communication language, it is not enough for them to work together in cooperative problem solving environments. Task sharing or task assignment between agents in a multiagent environment realized via some agent communication protocols. At this point it is important to state the difference between task execution cost, efficiency, capacity and protocol's run-time capacity and efficiency. Task execution cost, capacity or efficiency usually more related with the task's itself, its complexity etc. On the other hand, for example, if we consider the task sharing environment, completion of a task includes the assignment process of the task besides the execution process. In other words, with more efficient (or less inefficient) protocol, finding the participant to share the task is going to improve the overall performance.

Agent communication protocols can be predefined as Contract Net, English Auction, Dutch Auction, etc. or can be ad-hoc. Negotiations between the agents are realized by these protocols if they are in competitive environment. As it is stated in chapter two, higher channel capacity points to less inefficiency and less inactivity. It is obvious that for two protocols which are alternative of each other, less inefficient or less inactive one would be preferred. It is very important to underline that once again, proposed metric is a design time metric. Since agent cooperation in a multiagent system is a very complex process, it is valuable to have a clue about the run-time efficiency of the protocol during the design time. Protocol designer can model; his/her protocol and test it against various predefined communication protocols (if they are alternative to designed protocol). Or for any possible modifications in the protocol, designer can compare the existing protocol with the modified one by means of efficiency at design time. In other words, let's say one has designed two protocols that are alternative to each other and would like to know which one of these protocols is more efficient. Instead of developing his/her software with those protocols and testing them, using proposed metric, without doing all those steps can conclude that which protocol is more efficient.

In the following sections proposed metric is first going to be applied to the agent communication protocols by comparing Contract Net and Iterative Contract Net (which is a modified version of Contract Net).

3.1 Application of the Metric to Agent Communication Protocols

To illustrate the application of Shannon's combinatorial capacity calculation metric to the agent communication protocols we will use two well known protocols. First one is the Contract Net [8] and the other one is the Iterated Contract Net [9] which is the modified type of Contract Net.

3.1.1 FIPA Contract Net Communication Protocol

Contract Net protocol is developed to carry out negotiations between participants in distributed environments. Contract Net allows the distribution of tasks among a group of agents. eBay like auction marketplaces are good examples of Contract Net. A user in marketplace specifies his/her goods (call for proposal act) and among the bids received (proposal act) chooses the winner, and in the eBay example winner is the highest bidder.

In the contract net, one agent (the Initiator) takes the role of manager which wishes to have some task performed by one or more other agents (the Participants) and, further wishes to optimise a function that characterizes the task. This characteristic is commonly expressed as the price, in some domain specific way, but could also be soonest time to completion, fair distribution of tasks, etc.

As shown in Figure 3, initiator agent starts the communication sending *Call for Proposal* (CFP) to the participants in the environment. Participants can reply the CFP either with *refuse* or *propose* performative. Communication continues with the participants that replies with *propose* performative. After initiator agent collects proposals from the participants, can accept one of them and refuse the rest or refuse all of the proposals. If a proposal accepted then related participant informed with the *accept-proposal* act. After the given task is executed by the participant, initiator is going to be informed according to the result of the execution with *inform* or *failure* performative.

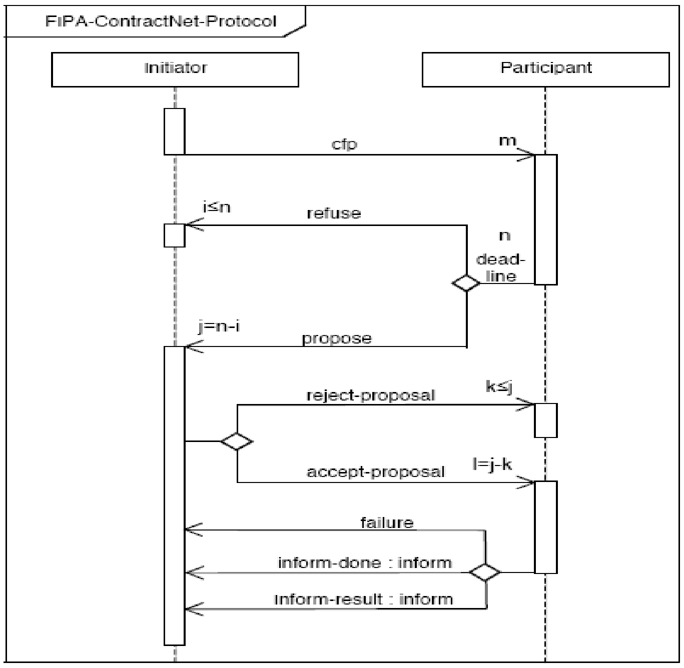


Figure 3 – Sequence Diagram for Contract Net

Graph representation of Contract Net protocol is as follows:

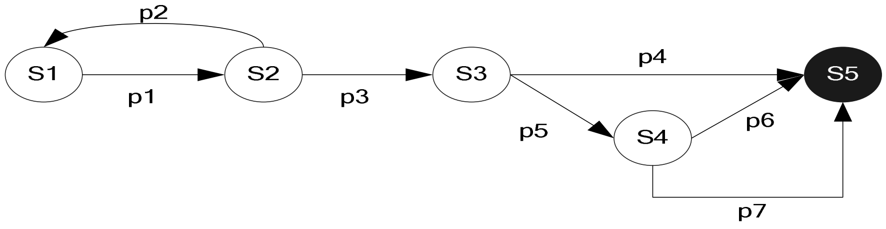


Figure 4 – Graph Representation of Contract Net Protocol

In Figure 4, S1 is the initial state where initiator agent sends the *Call for Proposal* performative. S2 is the state where participants reply call for proposal either by *refuse or propose* performative. In state S3, initiator either accepts or rejects one of the proposals sent by participants. If the proposal accepted by the initiator, in state S4 participant informs the initiator with the result. If the result is *done* then system goes to the final state S5, in the case of failure system also goes to final state S5 and communication ends. If proposal rejected by the initiator then initiator sends the reject performative to the participant and system goes to the final state S5. Performatives that realize the communication are labeled as p1 to p7. Description of the performatives can be found in Table 2.

As it mentioned before, ACL has more than 20 performatives. These performatives can be represented by two fundamental performative called ***inform*** and ***request***. From the definitions in Chapter 2, one can easily interpret that, graph representation of Contract Net protocol represents a Shannon language over alphabet A. Elements of alphabet A is the performatives that used to realize the protocol. As we mentioned in the previous section, task execution efficiency is dependent to the task assignment. So that, in such environment initiator agent interacts with many participants and gathers proposals from all of them to choose the best bid. It means that, contract net protocol is going to be executed many times concurrently. Therefore, capacity of contract net protocol is important since winning bidder is determined at the end of those communications which adds an overhead to the task execution efficiency.

Table 2. Performative Descriptions of Contract Net

Symbol	**Performative**
P1	Call for Proposal
P2	Refuse
P3	Propose
P4	Reject Proposal
P5	Accept Proposal
P6	Inform (Result/Done)
P7	Failure

One can, define the durations (or costs) of the performatives, based on the atomic performatives, ***inform*** and ***request,*** since each performative can be represented by these atomic performatives. In Table 3, you can find the description of FIPA ACL performatives based on the atomic performatives inform and request, and relative costs that we assign them to use in our calculations. In here, to differentiate the costs of inform and request, we assumed that inform has 1 unit cost and request has 2 unit costs.

Table 3. Description of Performatives based on inform and request

Performative	**Inform**	**Request**	**Cost**
Call for Proposal	1	2	3
Refuse	1	-	1
Propose	1	-	1
Reject Proposal	1	-	1
Accept Proposal	1	-	1
Inform	1	-	1
Failure	1	-	1

In this example, capacity, inefficiency and inactivity values of the contract net protocol calculated first by taking cost of each performative equal and one, then we will recalculate these values with the cost values in the Table 3. From Definition 10, branch duration partition function for equal costs is;

$$P(s) = \begin{bmatrix} 0 & e^{-s} & 0 & 0 & 0 \\ e^{-s} & 0 & e^{-s} & 0 & 0 \\ 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and for relative costs in Table 3, P(s) is;

$$P_1(s) = \begin{bmatrix} 0 & e^{-3s} & 0 & 0 & 0 \\ e^{-s} & 0 & e^{-s} & 0 & 0 \\ 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

From Theorem 1, Definition 11, Definition 12, information capacity, inefficiency and inactivity values of Contract Net protocol is shown in Table 4.

Table 4. Capacity, inefficiency and inactivity values of Contract Net Protocol for equal and inform-request based costs

	$C_{\text{comb (bits/sec)}}$	**ζ (inefficiency) (sec/bit)**	**Φ (inactivity) (sec/bit)**
$P\ (s)$	0.2406	4.156	0.8467
$P_1(s)$	0.1406	7,1123	0.9074

3.1.2 FIPA Iterated Contract Net Protocol

Iterated Contract Net Protocol is the modified version of the Contract Net Protocol. Different from the Contract Net protocol, negotiation is iterated fixed number of times by the initiator agent to get better bids. During the iterations, according to the proposals get initiator agent revises its call for proposal and starts the negotiation process over the participants replied with propose act in the previous iteration. In Figure 5, you can find the sequence diagram of Iterated Contract Net protocol. Graph representations of Iterated Contract Net protocols for two, three, and four iterations can be seen in Figure 5a, 5b and 5c.

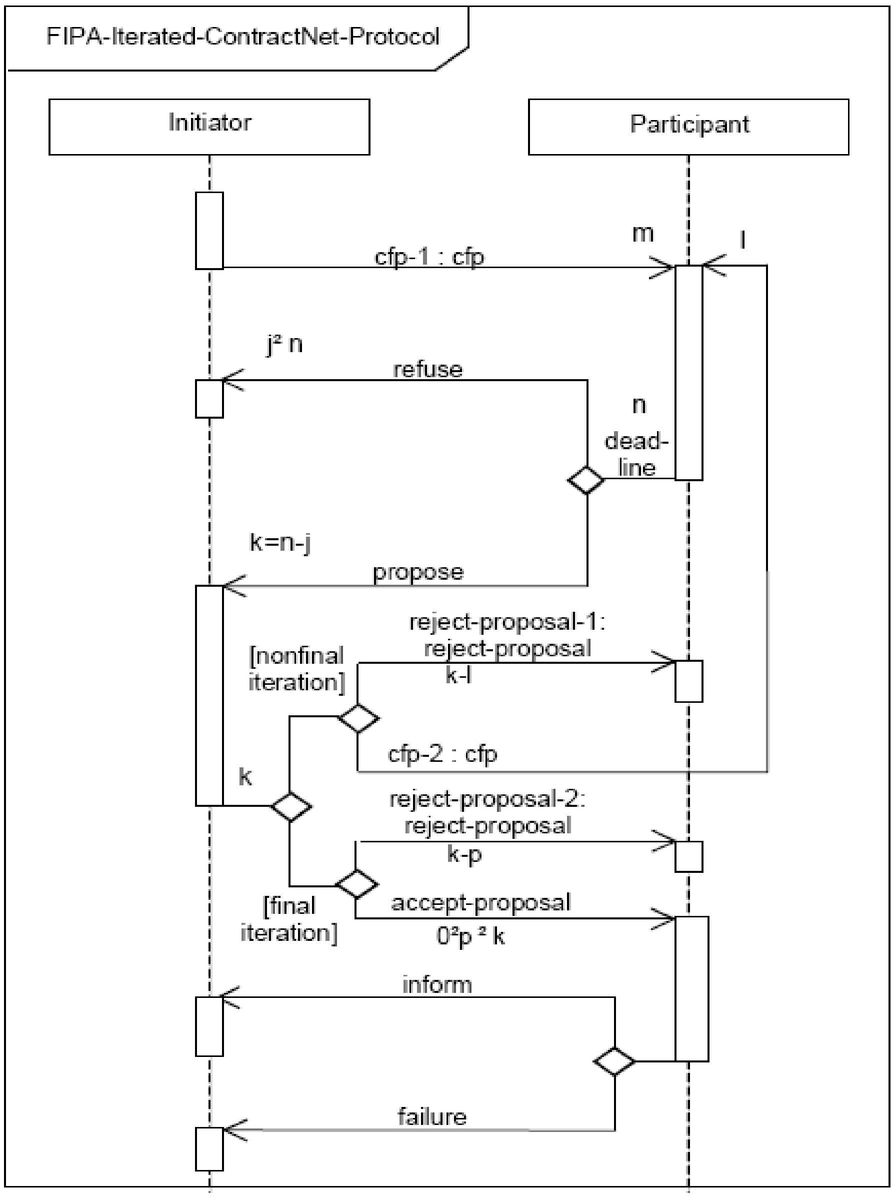


Figure 5 – Sequence Diagram of Iterated Contract Net

And, the performative that used in Iterated Contract Net can be seen in Table 3.

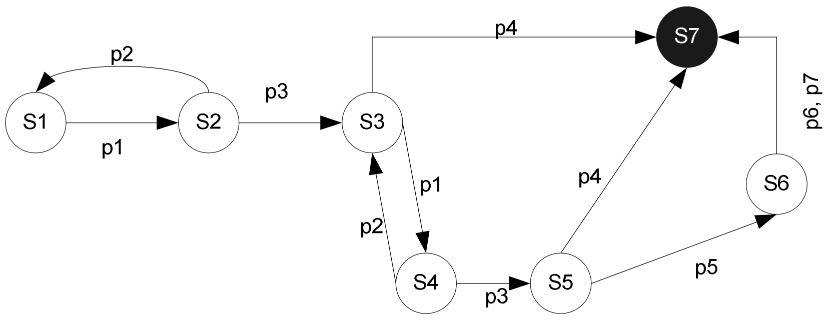


Figure 6.a – Graph Representation of Iterated Contract Net Protocol for 2 Iterations

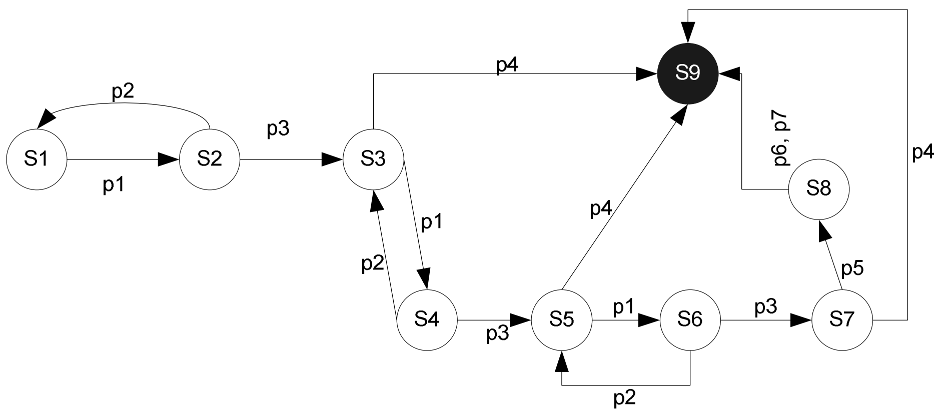


Figure 6.b – Graph Representation of Iterated Contract Net Protocol for 3 Iterations

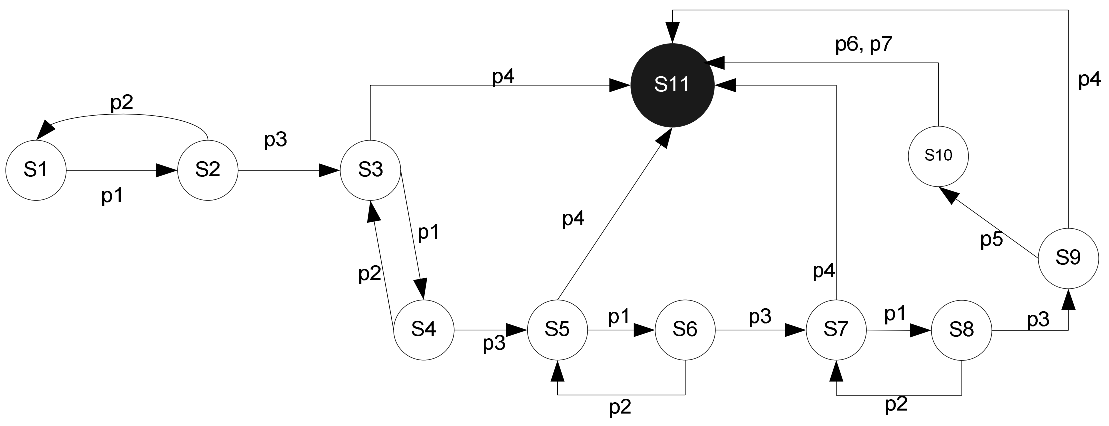


Figure 6.c – Graph Representation of Iterated Contract Net Protocol for 4 Iterations

From Definition 10, branch duration partition functions for two, three and four iterations of Iterated Contract Net protocol is shown below as P_1, P_2 and P_3 respectively.

$$P_2(s)=\begin{bmatrix} 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 \\ e^{-s} & 0 & e^{-s} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{-s} & 0 & 0 & e^{-s} \\ 0 & 0 & e^{-s} & 0 & e^{-s} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-2s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P_3(s)=\begin{bmatrix} 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e^{-s} & 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{-s} & 0 & 0 & 0 & 0 & e^{-s} \\ 0 & 0 & e^{-s} & 0 & e^{-s} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-s} & 0 & 0 & e^{-s} \\ 0 & 0 & 0 & 0 & e^{-s} & 0 & e^{-s} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-2s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P_4(s)=\begin{bmatrix} 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e^{-s} & 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 & 0 & e^{-s} \\ 0 & 0 & e^{-s} & 0 & e^{-s} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-s} & 0 & 0 & 0 & 0 & e^{-s} \\ 0 & 0 & 0 & 0 & e^{-s} & 0 & e^{-s} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-s} & 0 & 0 & e^{-s} \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-s} & 0 & e^{-s} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-s} & e^{-s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-2s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

From Theorem 1, Definition 11, Definition 12, information capacity, inefficiency and inactivity values of P_2, P_3 and P_4 are shown in Table 5.

Table 5. Capacity, inefficiency and inactivity values of Iterated Contract Net Protocol for k=2,3,4 Iterations

	$C_{\text{comb (bits/sec)}}$	**ζ (inefficiency) (sec/bit)**	**Φ (inactivity) (sec/bit)**
$P_2(s)$	0.2812	3.5561	0.8229
$P_3(s)$	0.2992	3.3422	0.8127
$P_4(s)$	0.3094	3.2320	0.8070

From the calculations above, it is seen that when the iteration number increases capacity of the protocol increase too. From that point, one can conclude that Iterative Contract net protocol is more efficient than Contract Net protocol especially if we think about the concurrent execution of the protocols in the multi agent environments.

3.2 Application of the Metric to Multiagent Communication Topologies

Multi Agent environments are virtual places where agents come together for problem solving. To achieve their goals in multiagent environments agents communicate with each other. At this point, it is important to state some important differences between object and agent paradigms. Objects have a centrally organization while agents allows distributed computing. An object class usually has a specific capability or functionality which would facilitate inheritance while agents could play different roles in different domains or situation. Object communication limited to one-to-one while agents allow multiple and parallel communication.

In this section we investigate the information capacity of the multiagent communication topologies due to the connectivity of the environment. We use “Small World” [16], “Random” [18] and “Hyper Grid” [17] topologies as sample multiagent communication topologies to evaluate the communication capacity of the multi agent systems.

It is better to give some explanations about the terminology such as diameter, clustering coefficient, random graph, wiring probability and baseline model which are going to be used to describe multiagent communication topologies. The diameter of a graph is the greatest distance between two nodes of the graph. Clustering coefficient is the measure assesses the degree to which nodes tend to cluster together. A random graph is obtained by starting with a set of n vertices and adding edges between them at random and generally used as baseline model. Wiring probability is the link making probability of one node with others. [19]

Nodes of the each topology are going to be the agents therefore it is important that no agent disappear before the communication ends because leave of any agent changes the connectivity. During the evaluations of the topologies, we will assume that agents in the environment are going to use two different communication protocols. According to the node identifiers in the topologies, if there exist a link between two nodes and the difference between their identifiers is odd then they will use the protocol "a" otherwise they will use protocol "b". For example, if there is a connection between nodes q1 and q2 then the arc connecting them will be labeled by *a*, on the other hand if there exist a connection between nodes q1 and q3 then the arc connecting them will be labeled by *b.* For both labels *a* and *b,* it is assumed that b = 2a.

In Chapter 2, definition 8, right resolving property is defined. According to the right resolving property, for each node in the graph (agents in this case) labels on the all branches should be unique, otherwise it makes the given Shannon Language non-deterministic. If an agent communicates with more than one agent using the same communication protocol, that conflicts with the right resolving property and we can not apply proposed metric because right resolving property is not satisfied in other words topology is non-deterministic. Therefore in order to apply proposed metric to multiagent topologies, we first need to satisfy right resolving property by making used topologies deterministic. To find the equivalent deterministic topology, a tool named JFLAP is used.

JFLAP is software for experimenting with formal languages topics including nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines, several types of grammars, parsing, and L-systems [24]. Accepting multiagent topologies as non-deterministic finite automaton, their equivalent deterministic finite automaton has been found by using NFA to DFA conversion of JFLAP. By nature, communication can start or end in any node (agent) of the topology. In other words, any node in the topology can be initial and final state at the same time. To apply the NFA to DFA conversion two dummy nodes are introduced as initial and final state, then these dummy nodes are connected to the others by empty moves labeled by the symbol λ.

3.2.1 Watts-Strogatz small world (WS-SW)

In small world network, there is a short path between any two nodes. The main characteristics of small world topologies are known to be low diameter and high clustering coefficient when we compare with random graph of equivalent size [16]. Neural networks, Voter networks, telephone call graphs and social influence networks are the real life examples of small world topology. In this part, in order to examine the communication capacity of WS-SW topology we will use three WS-SW generated graphs. Each of these graphs has eleven nodes and wiring probability of these graphs are 0.2, 0.4 and 0.8. All these three topologies generated via the IVC topology generator [11]. In Figures 7.a and 7.b, non-deterministic and deterministic form of the WS-SW topology with 0.2 wiring probability is shown. For the figures representing non-deterministic and deterministic form of WS-SW topologies having 0.4 and 0.8 wiring probability, please refer to appendix A.

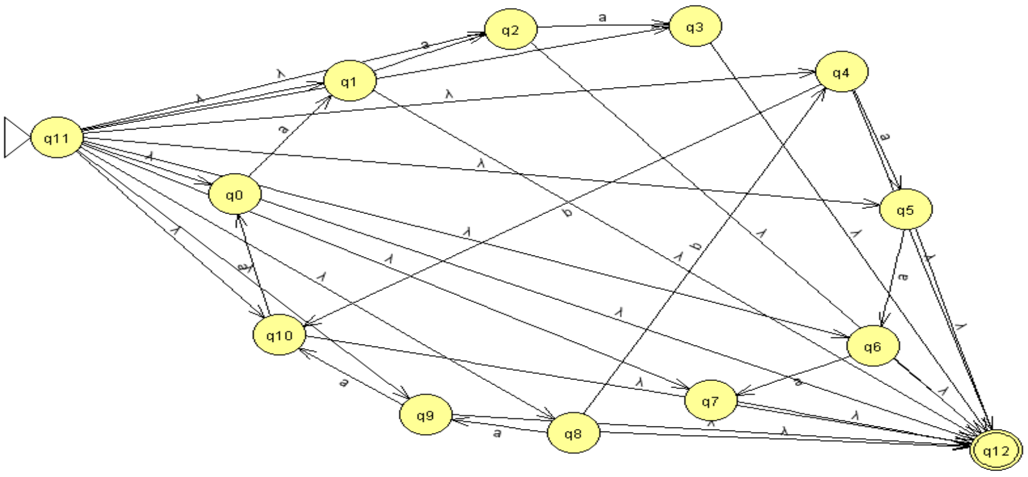


Figure 7.a – Non-Deterministic form of WS-SW with 11 nodes and 0.2 wiring probability

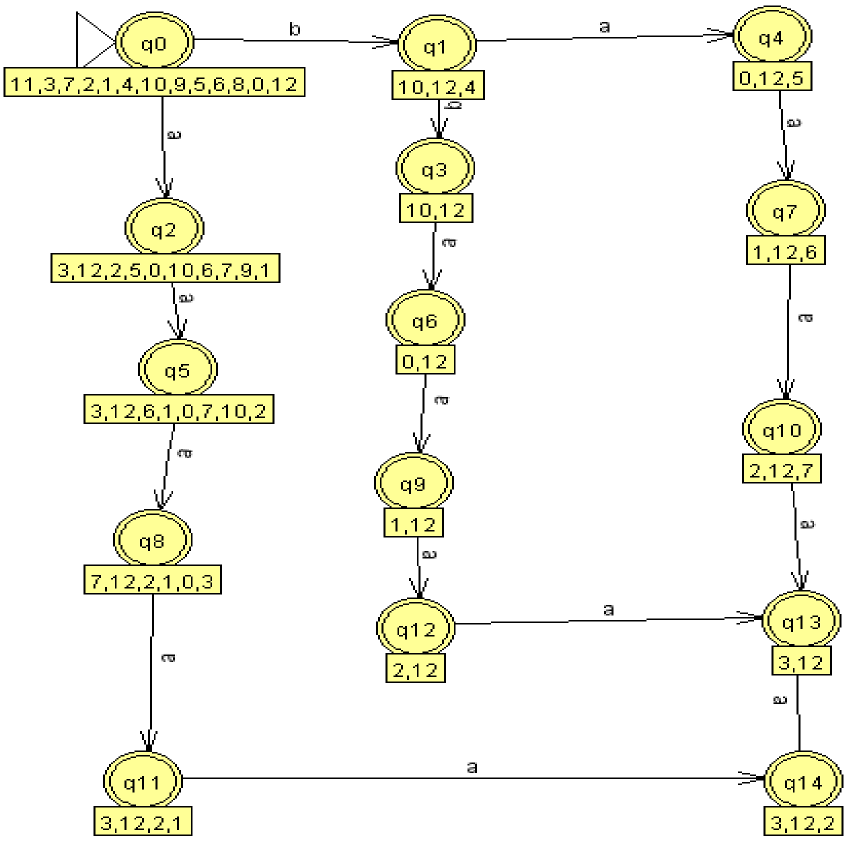


Figure 7.b – Deterministic form of WS-SW with 11 nodes and 0.2 wiring probability

3.2.2 Random Topology (RND)

Random graph model is utilized as a baseline model for comparison. In the model, the connection probability p is the only parameter used for topology building. The graph is constructed by adding an edge between any two nodes with probability p [10]. Each of these graphs has eleven nodes and wiring probability of these graphs are 0.2, 0.4 and 0.8. All these three topologies generated via the IVC topology generator [11]. In Figures 8.a and 8.b, non-deterministic and deterministic form of the RND topology with 0.2 wiring probability is shown. For the figures representing non-deterministic and deterministic form of RND topologies having 0.4 and 0.8 wiring probability, please refer to appendix A.

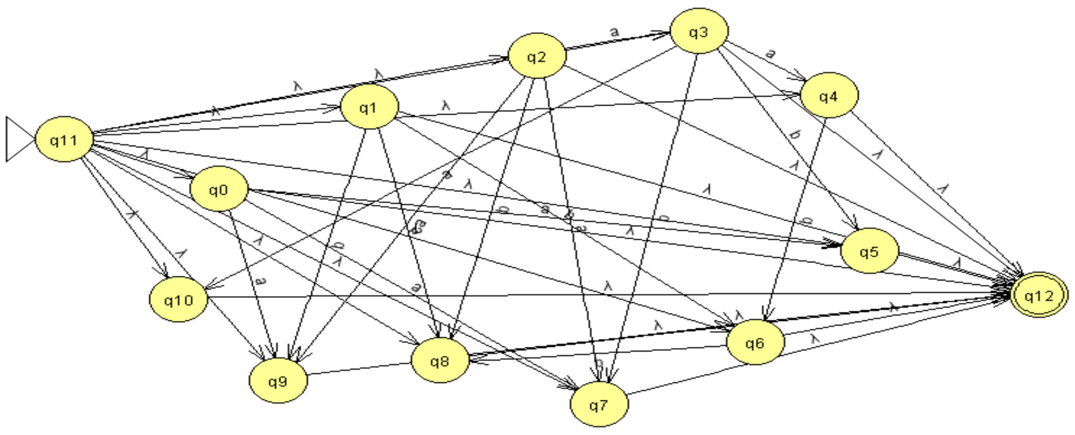


Figure 8.a – Non-Deterministic form of RND Topology with 11 nodes and 0.2 wiring probability

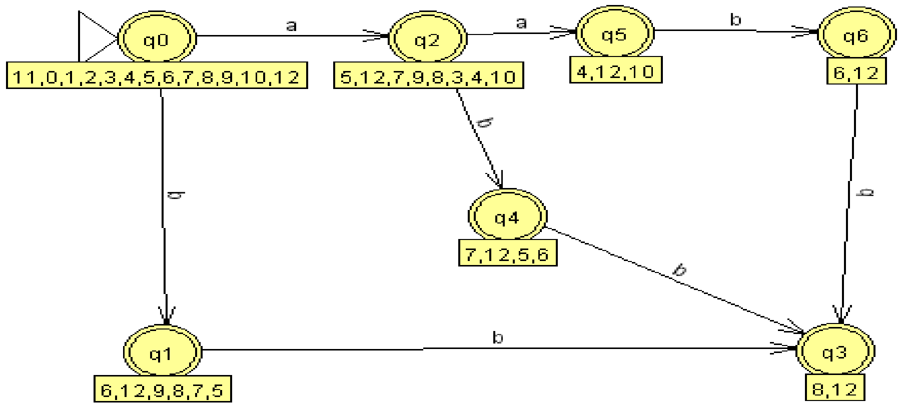


Figure 8.b – Deterministic form of RND Topology with 11 nodes and 0.2 wiring probability

3.2.3 HyperGrid (HG)

It is proposed by Saffre and GhaneaHercock [10]-[17] to model P2P networks. The main characteristic of the model is the generation of a graph topology with low graph diameter value and limited node degree. In Figures 9.a and 9.b, non-deterministic and deterministic form of the HG topology with 0.2 wiring probability is shown. For the figures representing non-deterministic and deterministic form of HG topologies having 0.4 and 0.8 wiring probability, please refer to appendix A.

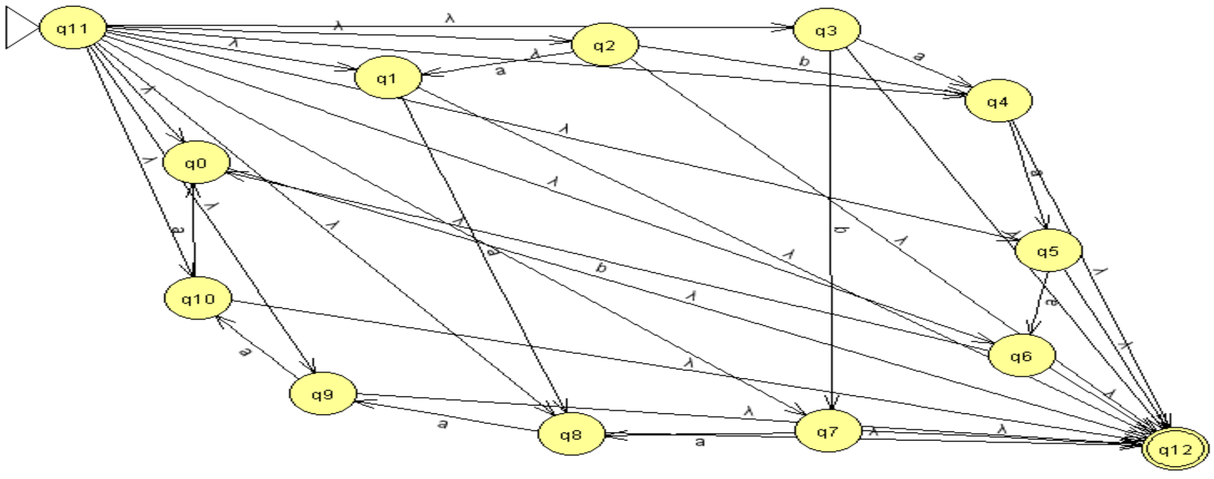


Figure 9.a – Non-Deterministic form of HG Topology with 11 nodes and 0.2 wiring probability

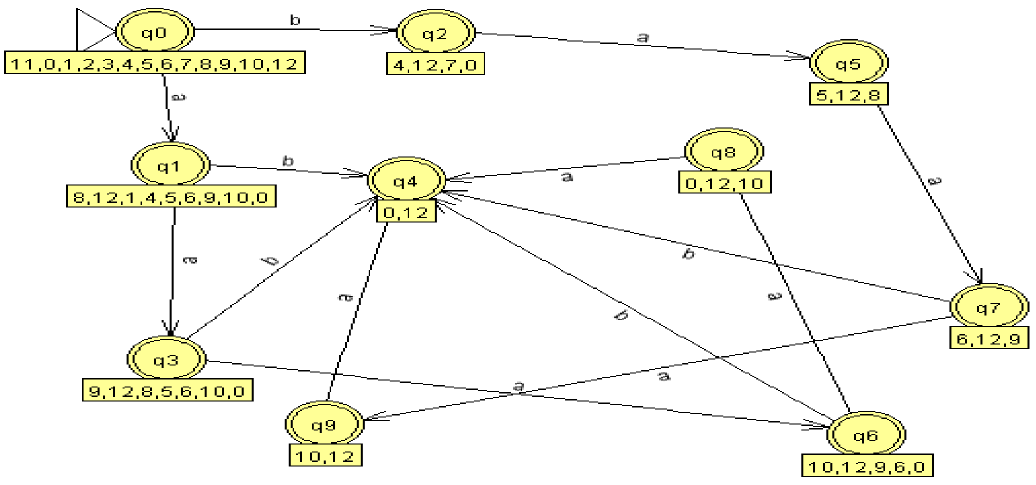


Figure 9.b – Deterministic form of HG Topology with 11 nodes and 0.2 wiring probability

The combinatorial capacities, inefficiency and inactivity metrics of each graph can be found in Table 6.

Table 6– Capacity values of multi agent topologies

Multi Agent Environment	C_{comb}			ζ (inefficiency)			Φ (inactivity)		
	WP = 0.2	WP = 0.4	WP = 0.8	WP = 0.2	WP = 0.4	WP = 0.8	WP = 0.2	WP = 0.4	WP = 0.8
WS - SW	1.0309	1.4742	2,8747	0.9700	0.6783	0.3479	0.4894	0.3599	0.1363
RND	0.1581	0.1960	0.4312	6.325	5.1020	2.3141	0.8962	0.8729	0.7416
HG	0.3025	0.4326	0.8435	3.3058	2.3117	1.1855	0.8108	0.7409	0.5573

One can easily say that from the results in Table 6, since wiring probability increases, information capacity between the agents in the environment increases too. Also results show that, connectivity between the agents in the topology affects the communication efficiency of the environment and for the same connectivities WS-SW topology seems to more efficient then the others. This experiment can also be expanded to classify multi agent environments between them. Such as how information capacity will be affected since the number of agents in the environment increase or which environment's information capacity would be higher for high number of agents. It is important to say that, since the agent number in the environment increase that will also increase the size of the branch duration partition matrix as well. That may cause some problems in the calculation. We have performed our calculations in MATLAB R14. An in our study we have observed that since agent number exceeds 25 agents, calculations takes too much time and for some connectivities MATLAB could not perform the calculation.

CHAPTER IV

APPLICATION OF PROPOSAL TO PETRI-NET MODELED SYSTEMS

4.1 Petri Nets

Petri Nets are graphical, mathematical tools for modeling, formal analysis, and design of discrete event systems. Petri Nets are being used for modeling of complex systems and can be applied to any area or system that can be described graphically, like flow charts [12]. Communication protocols, distributed software systems, data flow computing systems, formal languages, compiler and operating systems, local area networks are some application areas of Petri Nets.

A Petri Net is a particular bipartite digraph with three types of objects which are places, transitions, and directed arcs [12]. In mathematical graph theory, a bipartite graph is a graph whose vertices can be divided into two disjoint sets. Petri Nets as a bipartite graph, any two vertices of the graph cannot be linked without any transition between them. Places in a Petri Net, represent conditions and transitions represent events. We can differentiate places between them as input and output places. An input place is graphically defined as a place which has a directed arc pointed towards to a transition. Similarly an output place is graphically defined as a place which has an incoming directed arc from a transition. A transition (or an event) has finite number of input and output places which represents pre-conditions and post-conditions of an event respectively. Arcs can be labeled by positive integers which show the weight of the arc. In general, k-weighted arc can be interpreted as k parallel arcs. In order to examine the dynamic behavior of the modeled system, i.e. the illustration of state changes of the system, each place may hold none or positive number of tokens. Presence or absence of a token in a place indicates that, truth condition for that place either satisfied or not [12]. In a specific time instance, distribution of the tokens is called as marking and shows the current state of the designed system. On the other hand existence of the k-tokens in the place, can also be interpreted as k data sources available at that place at time *t*.

Formal definition of a Petri Net is as follows. A Petri Net is a 5 – tuple, PN=(P,T,I,O,M_0) where :

P = $\{p_1,p_2,\ldots,p_m\}$ is a finite set of places

T = $\{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions where, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$

I : (P x T) → N is input function where N is a set of positive integers

O: (T x P) → N is output function

M_0: P → N is initial marking.

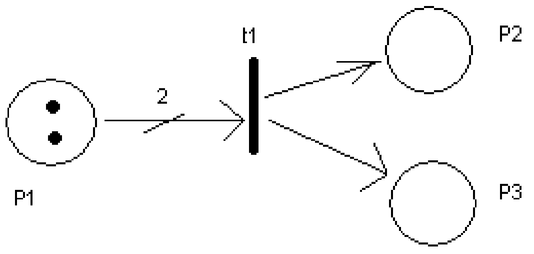
For many systems, the dynamic property of the system is described in terms of states and their changes. In order to illustrate the dynamic property of the modeled system states (or markings) are changed according to the firing rule:

1. A transition t is enabled, if each input place of t has at least w(p,t) tokens, where w(p,t) is the weight of the arc from p to t.

2. An enabled transition may or may not be fired depending on the event.

3. When a transition is enabled and fired, transition t removes tokens from its input places and distribute tokens to its output places according to w(t,p), where w(t,p) is the weight of the arc from t to p.

As shown in Figure 10, in order to fire transition t1, P1 should have at least two tokens, and when the transition t1 fires tokens are transmitted to the states P2 and P3.



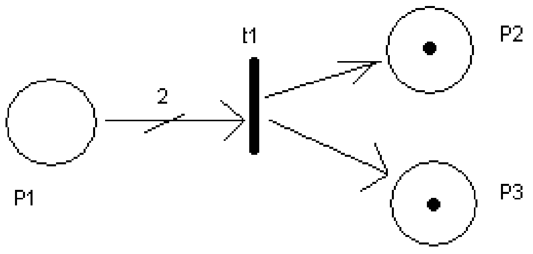


Figure 10.a. Petri Net before transition t1 fired.

Figure 10.b. Petri Net after transition t1 fired.

Petri Nets can be used to model any system that can be described as flow of events. For example," Finite State Machines are the subclasses of Petri Nets" [12].

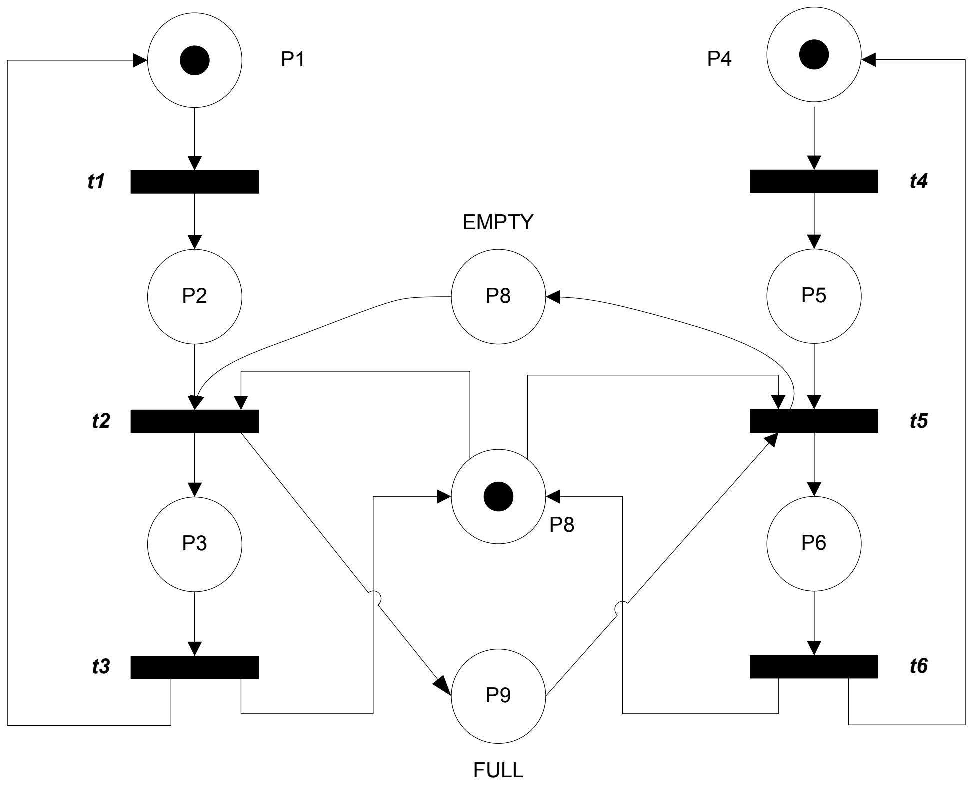


Figure 11. Petri Net Model of Robot Arm

In Figure 11, two robot arms perform pick and place operations by accessing a common workspace. To avoid the collision, only one arm can access the workspace at a time. In this model, places p1,p2,p3 and transitions t1,t2,t3 are belong to robot arm R1. Places p4,p5,p6 and transitions t4,t5,t6 are belong to robot arm R2. Transitions t1 and t4 represent concurrent activities of R1 and R2. Access to the workspace requires synchronization of the activities of the arms in order to avoid collision. This synchronization is done by the places p3,p6,p7 and transitions t2,t3,t5,t6. Firing transition t2 disables t5, assuming t5 is enabled, and vice versa. In addition, it is assumend that if p8 is empty, then t2 can not be enabled. This prevents R1 from attempting to transfer a part to the buffer since there is no space. Similarly, R2 cannot access buffer since there is no part in it, or place p9 is empty.

We can also use Petri Nets to represent formal languages [12]. If we give some symbolic labels (no need to be distinct) to the transitions of Petri Net, each transition sequence represents a string of symbols. The set of strings obtained by firing all possible transition sequences forms a formal language which is called a Petri Net Language in this case. On the other hand, since each finite state machine represents a regular language and every finite state machine can be represented as Petri Net, then every regular language can also be represented as Petri Net.

4.1.1 Properties of Petri Nets

In order to a Petri Net represent a Shannon Language, there has to be some restrictions on the design. First of all, designed system should be reachable in order to find out whether the modeled system can reach a specific state as a result of firing some set of transitions. Secondly, model should be bounded. Otherwise we can not guarantee that system would be in a unique state at any time t, which satisfies the right resolving property of Shannon language. Applicability of transition sequences through place visits without any deadlock occurs is important therefore we need liveness property of Petri Nets. Below some basic properties of Petri Nets are explained.

i. Reachability

From the perspective of modeling dynamic behavior of the system, reachability is a very important property. Before explaining the reachability, it is better to mention about what is marking. According to the firing rule, when a transition fired it changes the current marking scheme (distribution of tokens) of the net. If there exists a firing sequence that transforms initial marking M_0 to M_n then M_n is reachable from M_0. From here, we can say that, if all markings are reachable from initial marking M_0 by firing necessary transition rules (firing sequences) then Petri Net is reachable. Reachability property is very important because it tells us whether all requirements of system are met by Petri Net design or not. In other words, we can say that if all desired states are reachable by some sequence of transitions then the design satisfies all requirements of the system.

ii. Boundedness

A Petri net is said to be k-bounded if the number of tokens in any place p, is always less then or equal to k (where $k \geq 0$) for every marking M reachable from the M_0 . Therefore boundedness implies that the number of firing sequences of the designed net to be finite.

iii. Liveness

A Petri Net is said to be live if, no matter what marking has been reached from M_0, it is possible to ultimately fire any transition of the net by processing through some further firing sequence. This means that, Petri Net guarantees deadlock free operation, no matter what firing sequence is chosen. For complex and large systems verification of the liveness is impractical. Therefore it is better to use the level of liveness defined by [1].

A transition *t* in a Petri Net (N, M_0) is said to be:

- Dead (L0 – live) if *t* is not fired in any firing sequence.
- L1 – live if *t* is fired at least once in some firing sequence.
- L2 – live, if *t* is fired at least *k* times in some firing sequence.
- L3 – live, if *t* is appeared infinitely in some firing sequence.
- L4 – live, if *t* is L1 – live for every marking in the net.

In this work, only L4 – live nets are considered, because it is the most correspondent level of liveness.

iv. Reversibility and Home State

A Petri net is reversible, if for each marking M , M_0 is reachable. Therefore in a reversible net it is possible getting back to the starting state (initial marking). For many applications, returning back to the starting state generally is not the case. But sometimes, it would be nice returning back to some (home) state. Therefore apart of initial state, if M′ is the home state if, it is reachable from any marking M.

v. Coverability

A marking M is coverable $M'(p) > M(p)$. Coverability property is closely related with L1-liveness.

vi. Persistence

Petri Net is persistent if and only if, for any two transitions, firing one of them will not disable the other one.

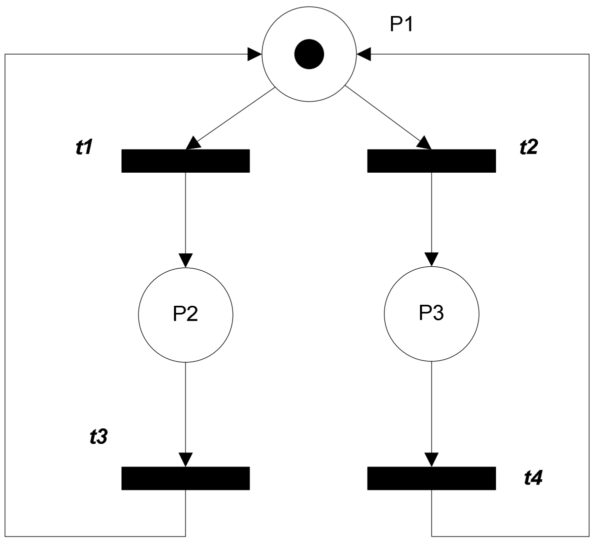


Figure 12 – Reachable, bounded, reversible, coverable and persistent Petri Net

Petri Net in Figure 12 is

- Reachable since all places are reachable by firing all possible transition sequences (t2 -> t4 and t3->t5).
- Bounded, because for all places number of tokens is finite.
- Live and coverable, since for every marking it is possible to fire a new transition.
- Reversible, since for places P2 and P3 there exists a transition that takes to system to the root place P1.
- Persistent since there is no transition that disables one another.

4.1.2 Analysis Methods of Petri Nets

There are three methods, related with the analysis of a Petri Net. First one is the Coverability Tree method, second one is the Matrix – Equitation method and the third one is the reduction or decomposition techniques. Techniques except the Coverability Tree method are out of scope of this work and, one can find detailed information about those techniques from [12].

Coverability Tree method is the method about firing all possible transitions of the given net. Starting from the initial marking M_0, one can obtain reachability set of the net by firing all transitions. In the coverability tree of the net, nodes are labeled with the markings starting from the initial marking M_0 and arcs are labeled by the transitions. If the net is not bounded, then coverability tree become indefinitely large. For unbounded nets, to keep the tree finite the symbol $\boldsymbol{\omega}$ introduced as infinite. Also, duplicate markings can make the reachability set large, in this case it would be necessary to extract the duplicates from the tree. Coverability tree is very important, because it shows all possible states of the system.

As it mentioned before, coverability tree of design could be unbounded. Since, boundedness of the Petri Net is required for representing it as Shannon language, instead of coverability tree, reachability graph of the Petri Net design has been considered as Shannon language. It is important to note that, extracting reachability graph of Petri Net from its coverability tree is possible if and only if coverability tree is k-bounded.

For large systems it is not really practical to extract Coverability tree from the Petri net. Also for calculating information capacity, if the number of places increases then computational complexity of capacity calculation increases and after some point it becomes impractical to calculate the capacity.

Following algorithm (taken from [3]) can be used to obtain Coverability tree of a Petri Net. In Figure 13.a extraction of coverability tree using the following algorithm of robot arm example can be seen.

```
1.0 Let the initial marking M0 be  the root of the tree and mark it "new"
2.0 While "new" markings exist do the following
3.0 Select a "new" marking M.
    3.1 If M is identical to the another marking in the tree, then tag M "old",
        and go another "new" marking.
    3.2 If no transitions are enabled in M than, tag M "terminal".
4.0 For every transition t enabled in marking M do the following:
    4.1 Obtain marking M' which results in firing t in M.
    4.2 If on the path from the root to M, there exist a marking M'' such that
        $M'(p) \geq M''(p)$ for each place p, and $M' \neq M''$, then replace M'(p) with
        $\omega$ for each p, wherever $M'(p) \geq M''(p)$.
    4.3 Introduce M' as a node, draw an arc from M to M' labeled t and tag it
        new.
```

In the figure below, you can see the Coverability tree of the Petri net in Figure 10 (Robot Arm). Since it is a bounded net (confirmed by using Petri Net Toolbox [25]), then we can extract reachability graph from its Coverability tree.

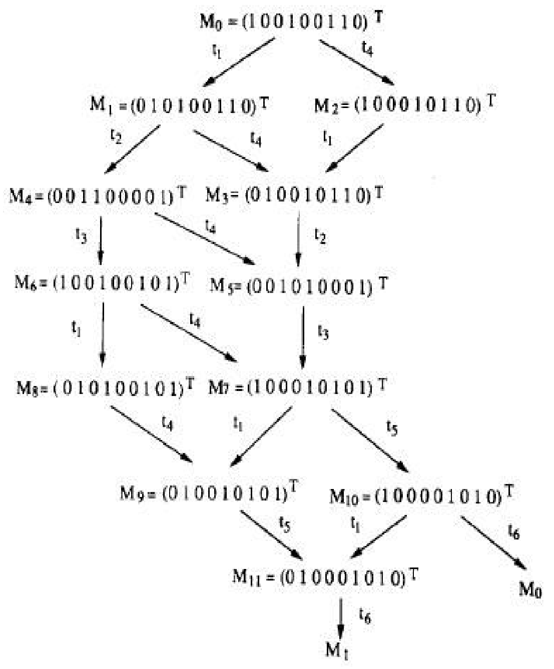


Figure 13.a – Coverability Tree of Robot Arm in Figure 11

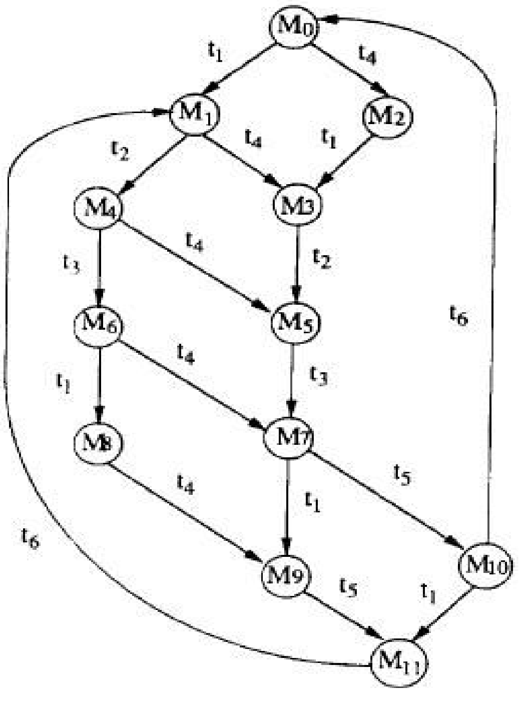


Figure 13.b – Reachability Graph of Robot Arm in Figure 11

4.2 Application of the Metric to Petri-Net Modeled Systems

Capacity of the Petri net is the maximum amount of information that can be processed through transitions of the Petri net, in per unit cost. The capacity of the Petri net can also be interpreted as the upper information processing limit of the design. On the other hand, one can intuitively conclude that the capacity of the Petri net also shows the constrainedness (or degree of freedom) of the designed system [13]-[14].

As it mentioned before, A Petri net is a bipartite digraph whose edges are labeled with transitions. From Definition 3 in Chapter 2, we know that Shannon language is also defined as a directed graph whose edges are labeled with the symbols from finite alphabet. To set up an analogy between Petri Net and Shannon language, given Petri Net should be k – bounded and deadlock free (or L4 – live). In our work we do not refer to the Petri net design directly but set up an analogy over its reachability graph where each marking represents global state of the system that are changed due to the transition events. From this point, the net should be k – bounded in order to obtain its reachability graph, also should be L4 – live since we need each marking reachable from M_0. Transitions of Petri net constitutes the alphabet of the Shannon language and associated costs can be interpreted according to the context of the design. In other words, it can be bit(s)/sec if the cost is determined as duration of the transition in time or resource consumption according to the resources allocated due to the firing of the transition sequences. Our proposed calculation procedure is as follows.

4.2.1 Capacity Calculation Algorithm

Inputs: A k-bounded,L4-live Petri Net design P. Transition cost $\tau(i)$ for each transition $t_{i.}$

Output: Combinatorial capacity of design (i.e. C_{comb}).

i. Obtain "marking reachability graph" G by using input P.

ii. Label the edges of G as Shannon alphabet letters by using transitions t_i and associated costs $\tau(i)$.

iii. Calculate combinatorial capacity C_{comb} of the constructed Shannon Language $L_{G,\lambda}$ using Theorem 1.

Example : Cruise Control System for Automobiles

Let us consider a cruise control system in Figure 14 with the following functionality. Cruise control is a system that automatically controls the speed of a motor vehicle. After the cruise control is turned the actual velocity of the vehicle can be stored with the SET-button and will be held constantly on this control value. Using the SET-button again the value of the velocity is incremented or decremented by 2 km/h. If the driver uses the brake of the vehicle the control of the velocity is suspended. It can be resumed (slider to RESUME and back to ON). In suspended state the actual velocity is compared with the stored control velocity and a buzzer is activated for one second if the control velocity is exceeded. The cruise control is turned off by pushing the slider to OFF [15]. Petri Net design of the cruise control system is shown in Figure 14.

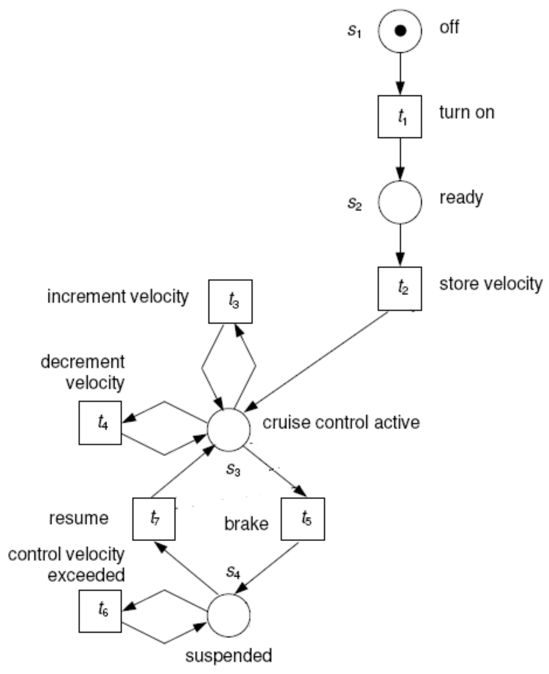


Figure 14. Petri Net Design of Cruise Control System

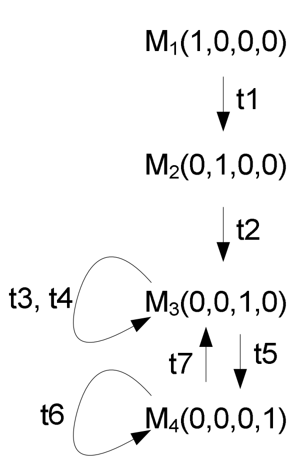


Figure 15.a Coverability Tree of Figure 14

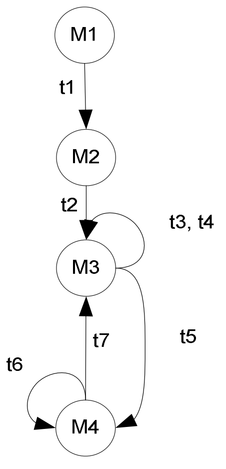


Figure 15.b. Reachability Graph of Figure 14

From the reachability graph of Figure 14 in Figure 15b design is reachable since all the places in the Petri Net design in Figure 14 exists in reachability graph. Also, since the number of tokens in the net is finitie design is bounded and there is no firing sequence that puts the system into a deadlock.

Designer of the Cruise Control System would like to modify his design in order to add distance warning functionality. With this functionality, cruise control system will warn the driver when the minimum distance between two vehicles exceeded and automatically decelerate the velocity. Designer by using the proposed metric can have an idea how the efficiency of the system is going to be affected due to this modification. In figure 16, the modified version of Cruise Control System is seen.

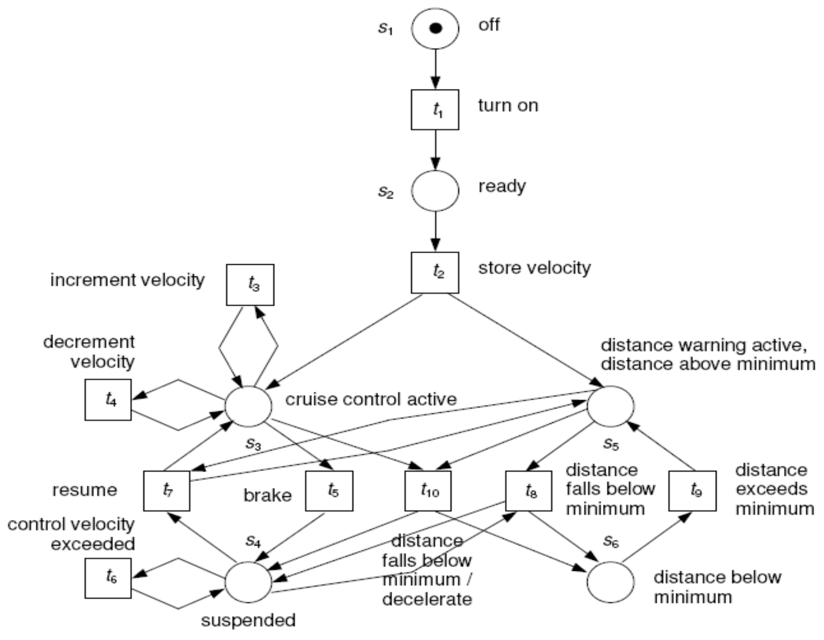


Figure 16. Petri Net Design of Modified Cruise Control System

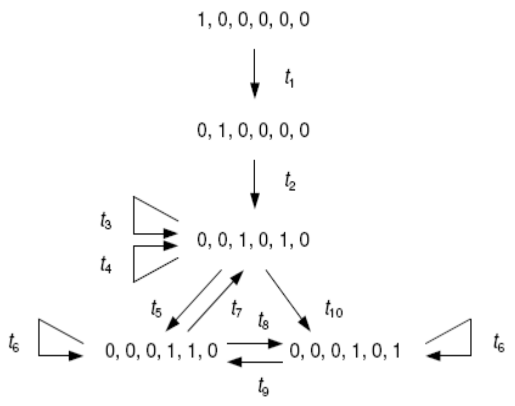


Figure 17.a Coverability Tree of Figure 16

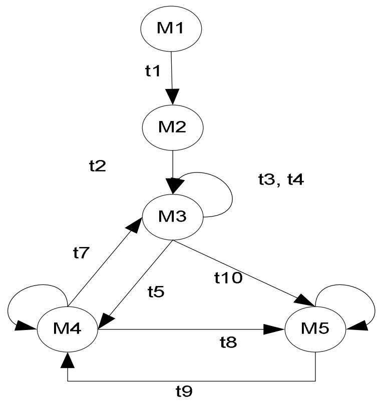


Figure 17.b. Reachability Graph of Figure 16

In figures 14 and 16, Petri Net designs of the Cruise Control System and its modified version with distance warning functionality are seen. Reachability graphs of both designs are shown in figure 15.b and 17.b. Reachability, boundedness and deadlock free properties of both designes are tested and verified by using Petri Net Toolbox [25]. As shown in both figures 15.b and 17.b reachability graphs represent a Shannon Language L, since places are the global states of the system and words of the L is obtained by reading-off the labels of transition sequences. Therefore both reachability graphs in figures 15.b and 17.b can be accepted as Shannon language and proposed metric can be applied to evaluate how modification affected the capacity, inefficiency and inactivity values of the system.

Table 7 Capacity Values of Cruise Control and Modified Cruise Control System

	Capacity (bits/sec)	Inefficiency (secs/bit)	Inactivity
Cruise Control	0.5889	1.6980	0.6648
Modified Cruise Control	1.0701	0.9344	0.4764

In Table 7; capacity, inefficiency and inactivity values of cruise control and modified cruise control system, in figure 14 and 16, is shown. Increase in the capacity value is expected, since a new functionality added to the system. In other words, new functionality naturally is going to increase the amount of information that needed by the system. Critical point is increase in the capacity shouldn't reach a peak point otherwise. In other words, it should not explode the information need of the system.

CHAPTER V

CONCLUSION

Application of Combinatorial Capacity Calculation metric of Shannon's Information Theory to Multiagent Software systems and Petri Net modeled software systems is the main theme of this thesis study. In this thesis, a metric is proposed, to measure the maximum runtime information capacity of software systems in the design time. Proposed metric is based on Shannon's Combinatorial Capacity Calculation metric and can be applied to any software system that can be described as Shannon Language. To show the application of proposed metric to the different areas of software engineering, Multiagent systems and Petri Net modeled software systems are used.

From the system designer's perspective, applications of proposed metric to the Multiagent systems are important. A multiagent system designer, by using this metric can evaluate his/her agent communication protocol design among other alternative designs to choose the best alternative by using inefficiency and inactivity values of the metric. Also, proposed metric can be used to evaluate the information capacity of the multiagent communication topologies. In order to apply the proposed metric to multiagent communication topologies, non-deterministic to deterministic conversion algorithm is used. According to the size of the designed system, this conversion can cause the state explosion problem which also increases the computational complexity of the capacity calculation.

To enhance the application area of the proposed metric, Petri Net modeling technique has been used. Petri Nets are well known modeling tools used in various engineering disciplines. There are lots of work exists in the literature about the modeling software systems by using Petri Nets. Proposed metric can be applied to any k-bounded, reachable and deadlock free Petri Net design. However it is important to note that analogy between Shannon Language and Petri Nets has been set up not directly with the Petri Net design but its marking reachability graph since each marking represents global state of the system that are changed due to transition events.

All the calculations in this work have been done by using symbolic toolbox of the R2006b version of the MATLAB. During the calculations it is noticed that, since the size of the design increases, computation of the capacity value consumes too much time and for some cases MATLAB can not compute the calculations. Therefore, for large systems it is impractical to use this metric.

In conclusion, proposed metric can be used by software system designers in order to evaluate their design among other alternatives. Also, system designer can evaluate the effects of the possible changes or modifications over the design by means of cost-effectiveness and performance with the inefficiency and inactivity values of the metric. Information Capacity value of the proposed metric does not give a certain idea about the performance of the system but with the use of the metric a common sense could be developed. Computational complexity of the proposed metric is the main obstacle about the usage of this metric and improvements can be done on it as a future work. Also applicability of proposed metric to the Petri Net area is an opportunity for other engineering disciplines. Therefore study about the applicability of this metric to other engineering areas can be considered as one another future work. Service Oriented Architectures (SOA) can also be the application area of this metric. Evaluation of the capacity values by means of the service quality can be interesting.

REFERENCES

[1] Şeker, R. and Tanik M. M.," An Information – Theoritical Framework for Modeling Component Based Systems". IEEE Trans. On Systems, Man and Cybernetics Part C: Applications and Reviews, Vol. 34, Issue 4, November 2004, pp 475-484

[2] Durfee, E. and Lesser, V. "Negotiating Task Decomposition and Allocation Using Partial Global Planning. In Distributed Artificial Intelligence, Volume 2, pp 229-244, Morgan Freeman

[3] Zurawski, R. and Zhou, M.C., "Petri Nets and Industrial Applications: A Tutorial", IEEE Transactions on Industrial Electronics, Vol. 41, No.6, December 1994, pp 567-583.

[4] Shannon, C. E., "A mathematical Theory of Communication". Bell System Tech. J., Vol. 27, pp. 379- 423, July 1948.

[5] Khandekar, A. McEliece, R. and Rodemich, E., "The Discrete Noiseless Channel Revisited". In Coding, Communications, and broadcasting, P. Farrel, M. Darnell, and B. Honary, Eds. Baldock, UK: Research Studies Series Ltd, 2000, pp. 115 – 137.

[6] Foundation of Physical Intelligent Agents : www.fipa.org , Last Visit: July 2009.

[7] FIPA ACL Specifications http://www.fipa.org/repository/aclspecs.html, Last Visit: July 2009.

[8] FIPA Contract Net Interaction Protocol, SC00029H

[9] FIPA Iterated Contract Net Interaction Protocol, SC00030H

[10] Cakir, B. and Kilic, H. “An Investigation about Process Matchmaking Performances of Unstructured and Decenterilized Digital Environments”, in IEEE DEST’07: Proceedings of the Inagural IEEE Intl. Digital Ecosystems and Technologies Conference, 2007.

[11] IVC Software : http://iv.slis.indiana.edu/sw/index.html . Last Visit: July 2009

[12] Murata, T. “Petri Nets: Properties, Analysis and Applications”. Proceedings of IEEE, Vol. 77, No. 4, April 1989, pp. 541-580

[13] de Assis, F.M. and Barros, T.C., “Evaluation of Petri Nets Compressibility”. Proceedings of International Symposium on Information Theory (ISIT), pp. 323, 1997.

[14] Gurajo, E.C., de Assis, F.M., Perkusich, A and Pimentel, C., “Petri Nets Compressibility and Capacity of Concurrent Systems”. Proceedings of International Symposium on Information Theory (ISIT), pp. 250, 2003.

[15] Gold, R. “Petri Nets in Software Engineering” Heft Nr. 5 aus der Reihe “Working Papers” June 2004

[16] Watts, D.J.; Strogatz, S.H. (1998). "Collective dynamics of 'small-world' networks.". Nature 393 (6684): 409–10.

[17] F. Saffre and R. G. Hercock, “Beyond anarchy: self-organized topology for peer topeer networks,” *Complexity*, vol. 9, pp. 49–53, November-December 2003.

[18] Erdős, P.; Rényi, A. (1959). "On Random Graphs. I.". Publicationes Mathematica 6: 290–297

[19] http://en.wikipedia.org/wiki/Distance_(graph_theory) , Last Visit: November 2009

[20] Kılıç, H.; Öztoprak, K. “Protocol and Connectivity Based Overlay Level Capacity Calculation of P2P Networks”, IEEE/WIC/ACM International Workshop on P2P Computing and Autonomous Agents (IEEE/WIC/ACM P2PAA 2006) in conjunction with the International IEEE WI/IAT'06 Conference, pp. 447-450, December 2006

[21] Hamou-Lhadj, A. “Measuring the Complexity of Traces Using Shannon Entropy”, Proceedings of the Fifth International Conference on Information Technology: New Generations, pp. 484-494, 2008

[22] Chapin, N., “Entropy-metric for systems with COTS software”, Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on Software Metrics, pp. 173 – 181, 2002

[23] Pimentel, C. and Bartolomeu, F. "A Combinatorial Approach to Finding the Capacity of the Discrete Noiseless Channel",IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 49, NO.8, AUGUST 2003

[24] JFLAP TOOL , http://www.jflap.org , Last Visit : November 2009

[25] Matcovschi, M. Mahulea, C. Pastravanu, O. "Tutorial : Learning about Petri Net Toolbox for use with MATLAB", Technical University Gh Asachi, 2005

APPENDIX

LIST OF FIGURES OF MULTIAGENT COMMUNICATION PROTOCOLS

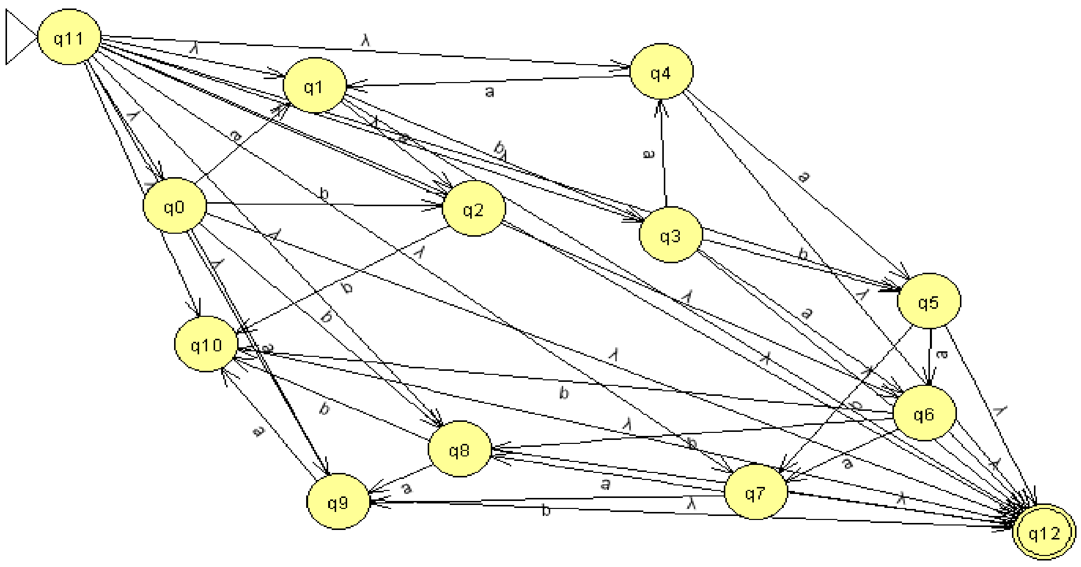


Figure A1 – WS-SW with 11 nodes and 0.4 wiring probability

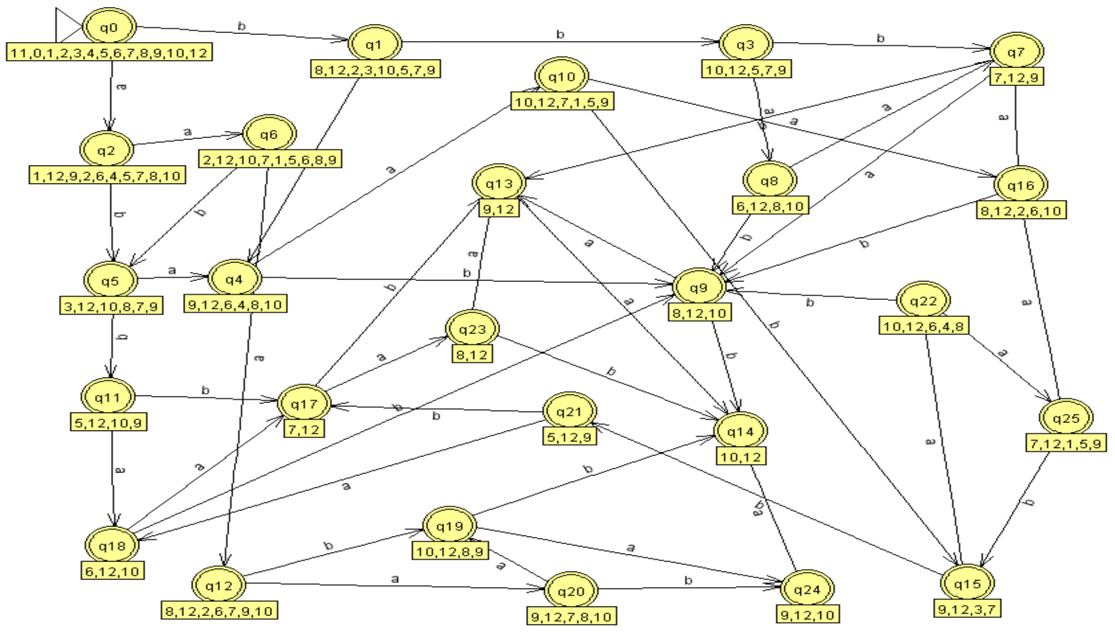


Figure A2 – Equivalent DFA of Figure A1

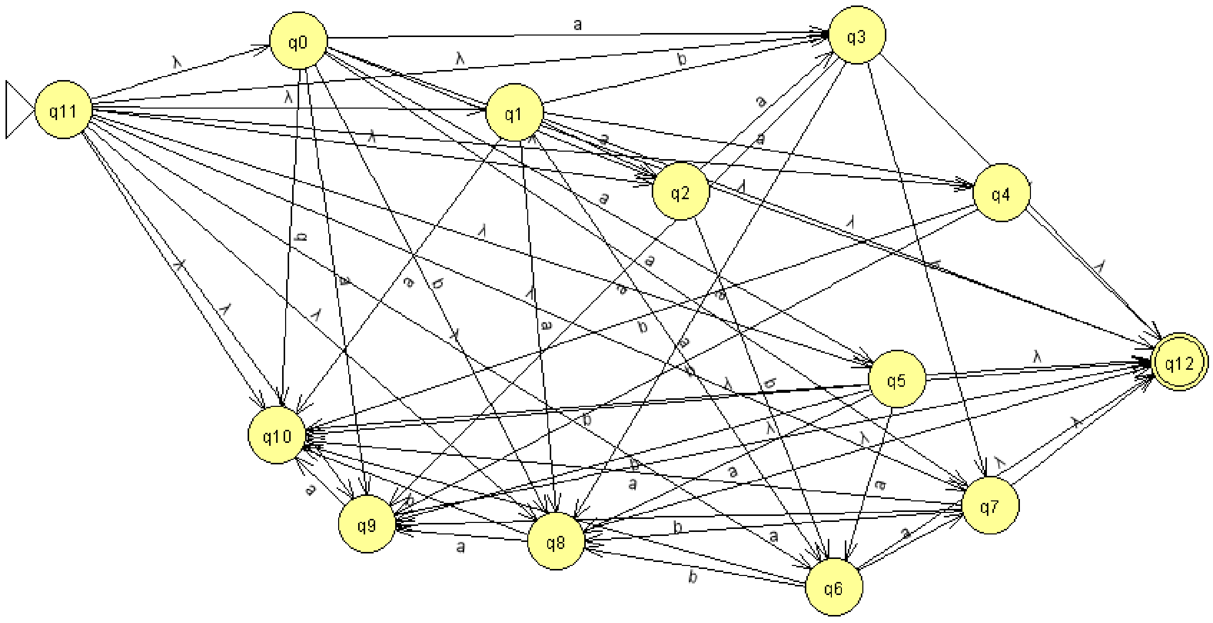


Figure A3 – RND Topology with 11 nodes and 0.4 wiring probability

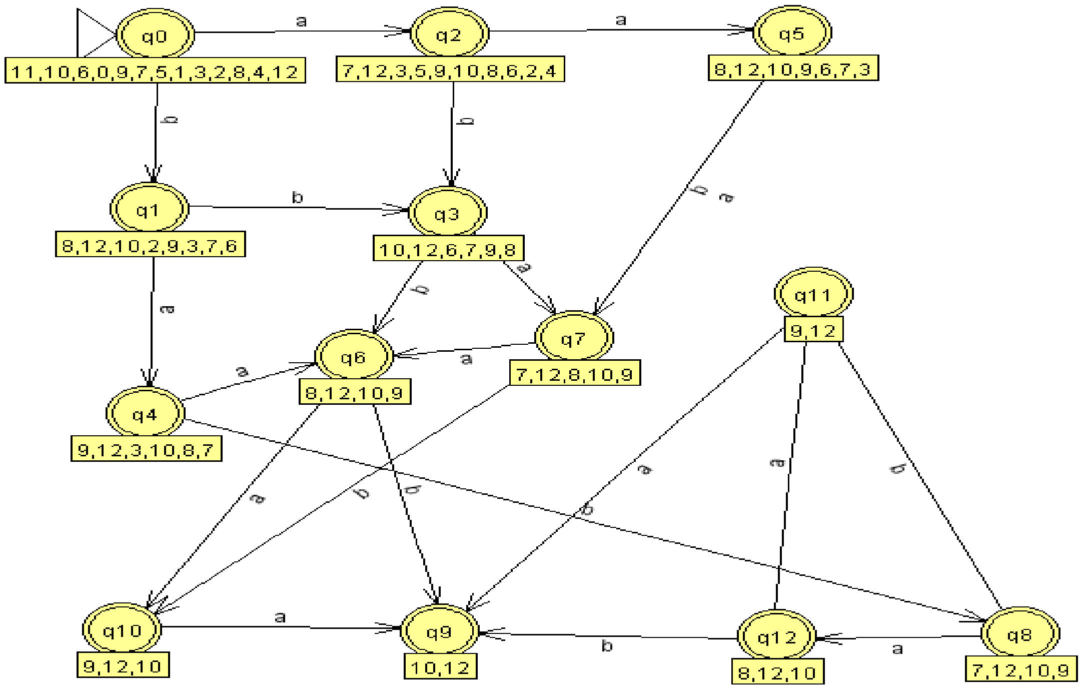


Figure A4– Equivalent DFA of Figure A3

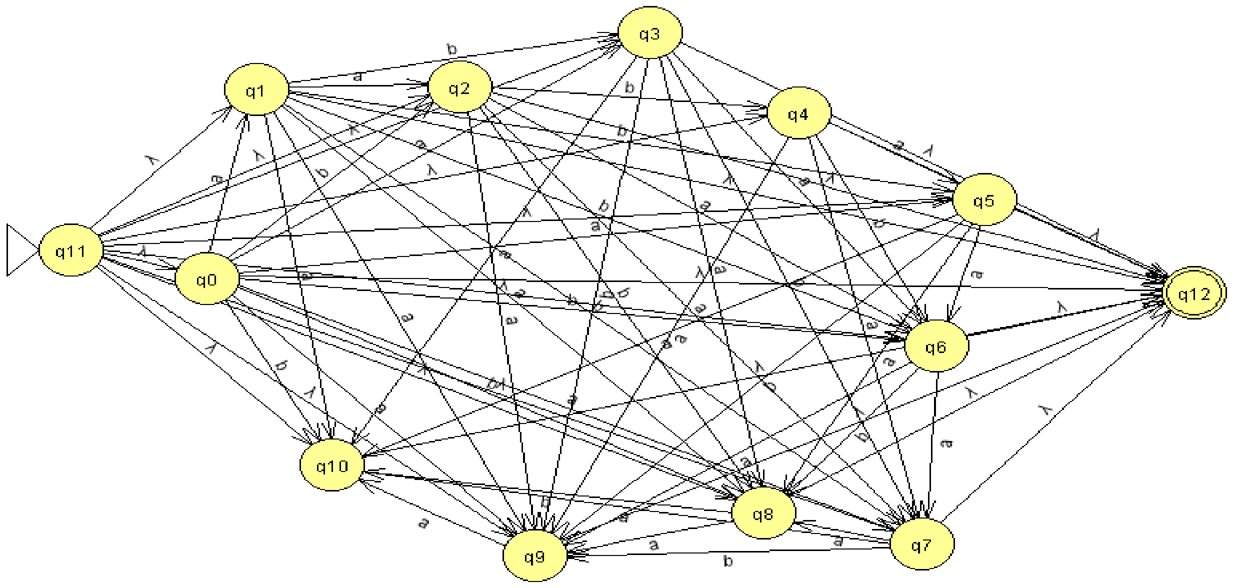


Figure A5 – RND Topology with 11 nodes and 0.8 wiring probability

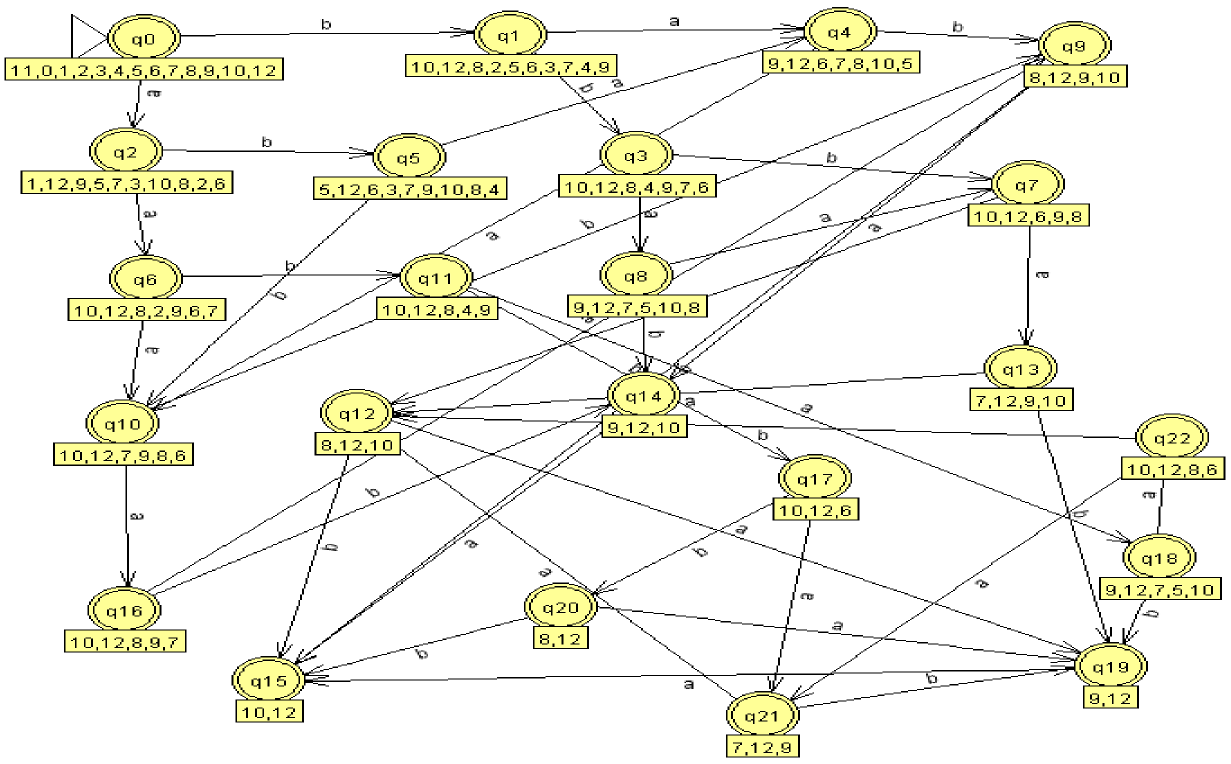


Figure A6 – Equivalent DFA of Figure A5

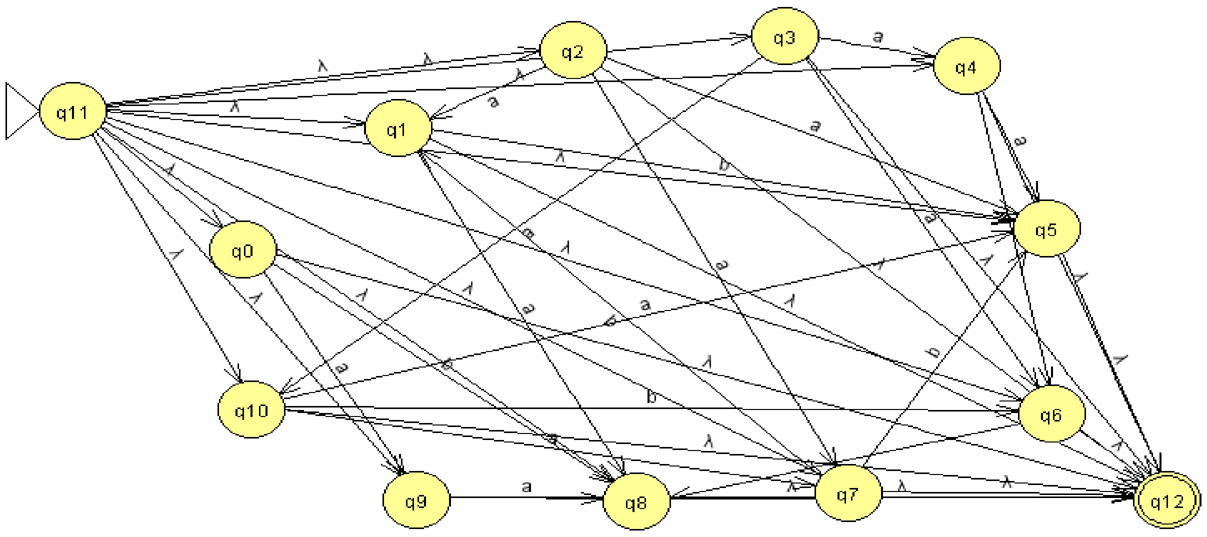


Figure A7 – HG Topology with 11 nodes and 0.4 wiring probability

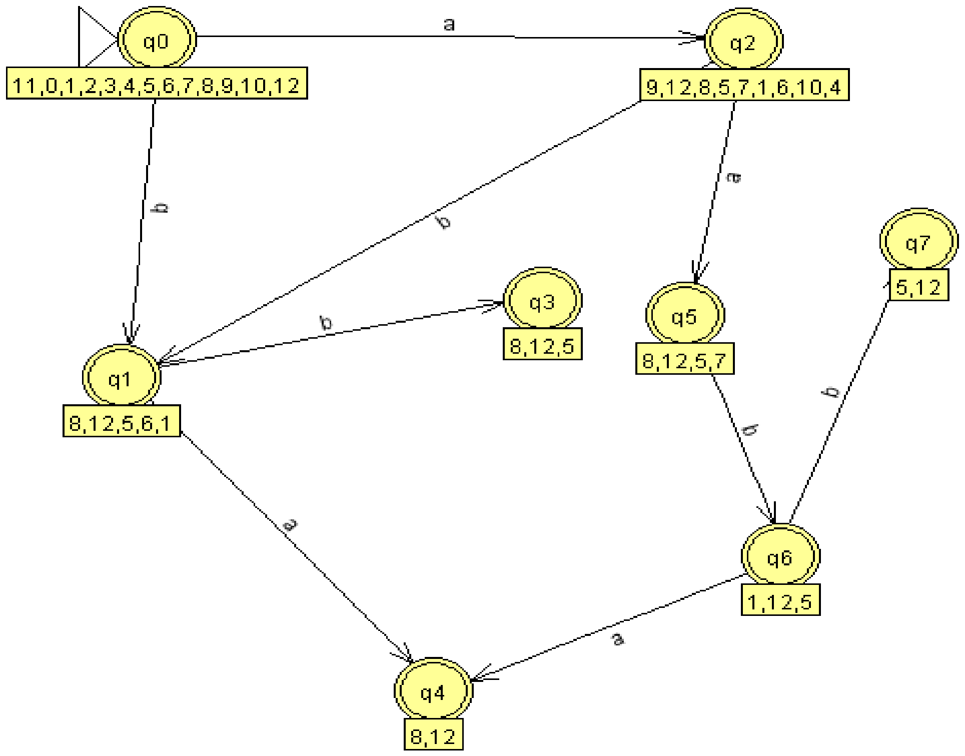


Figure A8 – Equivalent DFA of Figure A7

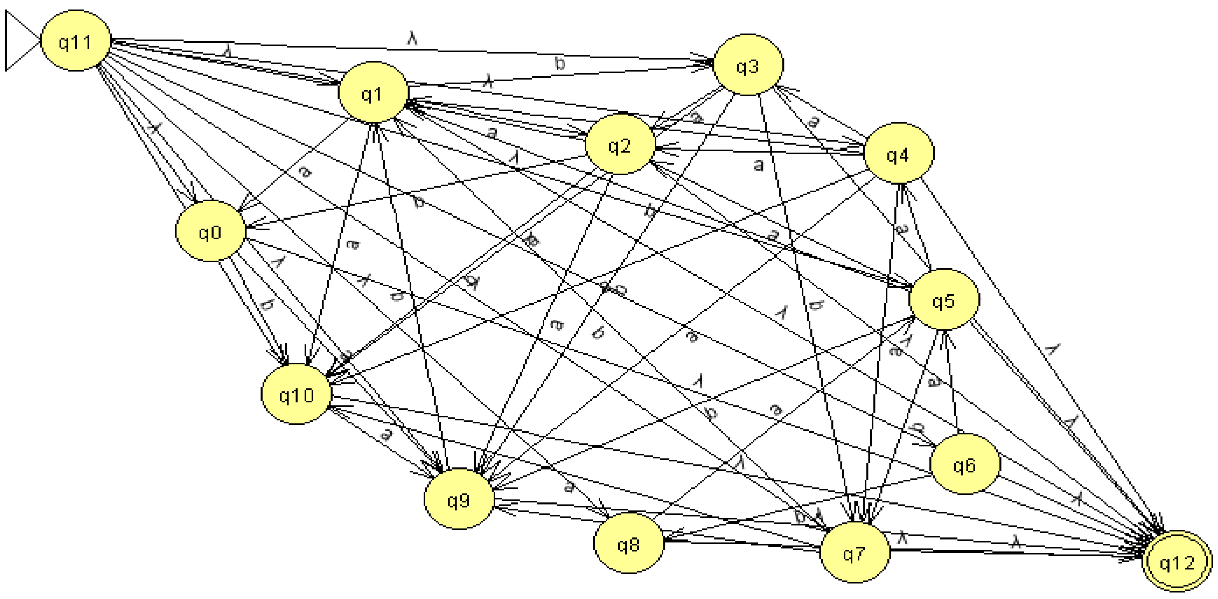


Figure A9 – HG Topology with 11 nodes and 0.8 wiring probability

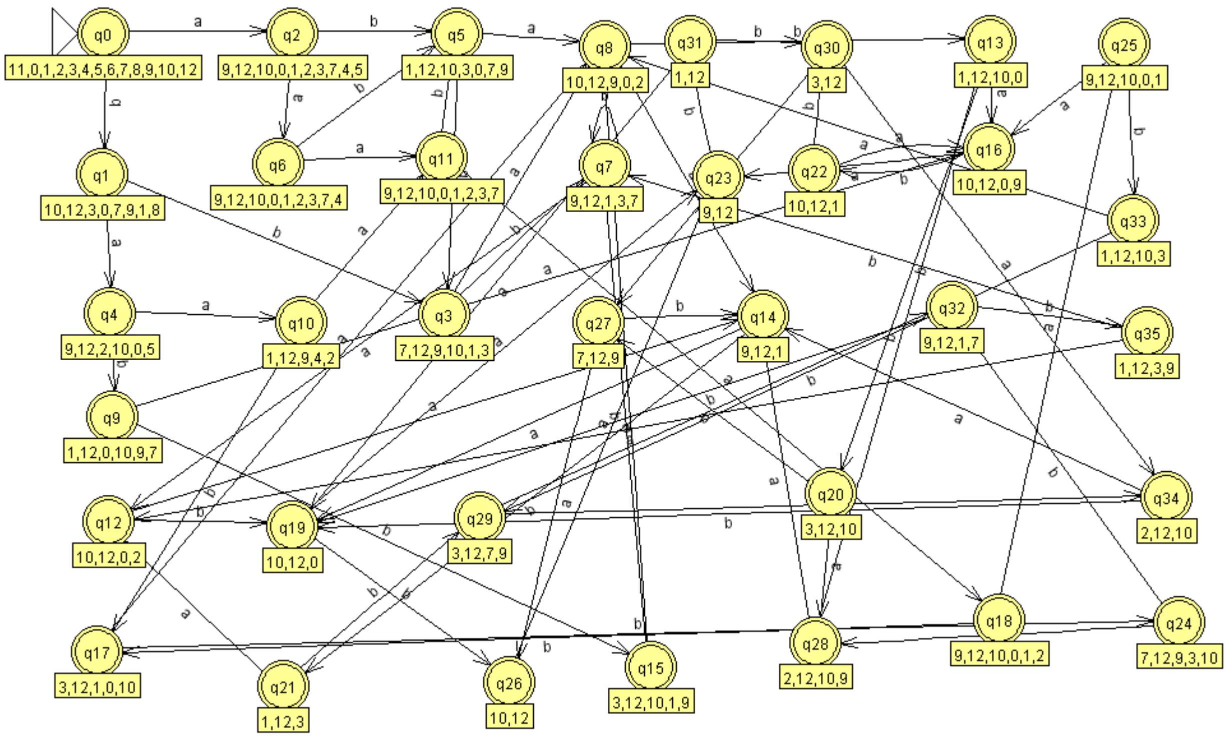


Figure A10 – Equivalent DFA of Figure A9