Issues of Real Time Information Retrieval in Large, Dynamic and Heterogeneous Search Spaces

John Korah

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

In

Computer Science And Applications

Eunice E. Santos, Chair
Calvin J. Ribbens
James D. Arthur
Jeffrey Borggaard
Eugene Santos Jr.

November 3rd 2009

Blacksburg, Virginia, USA

Keywords: Real time Information Retrieval, Large and Dynamic Search, Anytime Image Retrieval, Multiagent Resource Allocation Framework, Performance Analysis

Issues of Real Time Information Retrieval in Large, Dynamic and Heterogeneous Search Spaces

John Korah

Abstract

Increasing size and prevalence of real time information have become important characteristics of databases found on the internet. Due to changing information, the relevancy ranking of the search results also changes. Current methods in information retrieval, which are based on offline indexing, are not efficient in such dynamic search spaces and cannot quickly provide the most current results. Due to the explosive growth of the internet, stove-piped approaches for dealing with dynamism by simply employing large computational resources are ultimately not scalable. A new processing methodology that incorporates intelligent resource allocation strategies is required. Also, modeling the dynamism in the search space in real time is essential for effective resource allocation.

In order to support multi-grained dynamic resource allocation, we propose to use a partial processing approach that uses anytime algorithms to process the documents in multiple steps. At each successive step, a more accurate approximation of the final similarity values of the documents is produced. Resource allocation algorithm use these partial results to select documents for processing, decide on the number of processing steps and the computation time allocated for each step. We validate the processing paradigm by demonstrating its viability with image documents. We design an anytime image algorithm that uses a combination of wavelet transforms and machine learning techniques to map low level visual features to higher level concepts. Experimental validation is done by implementing the image algorithm within an established multiagent information retrieval framework called I-FGM. We also formulate a multi-

agent resource allocation framework for design and performance analysis of resource allocation with partial processing. A key aspect of the framework is modeling changes in the search space as external and internal dynamism using a grid-based search space model. The search space model divides the documents or candidates into groups based on its partial-value and portion processed. Hence the changes in the search space can be effectively represented in the search space model as flow of agents and candidates between the grids. Using comparative experimental studies and detailed statistical analysis we validate the search space model and demonstrate the effectiveness of the resource allocation framework.

Acknowledgements

I would like to thank my adviser Dr. Eunice Santos, without whose constant support and encouragement this work would not have been possible. She has inspired me with her ideals on scholarship, integrity and research that I hope to inculcate. I have enjoyed our discussions over a wide variety of topics ranging from philosophy to food. Due to her mentoring, I think I am better prepared to face the next set of challenges in my research career.

I would like to thank Dr. Eugene Santos Jr. for keeping me focused on the research topic. The insightful discussions that I had with him helped in refining the work presented in this dissertation. I would also like to thank Dr. Calvin Ribbens, Dr. James Arthur and Dr. Jeff Borggaard for agreeing to serve on my committee and for their helpful suggestions. Their constant support has been crucial for the successful completion of my graduate studies. I would also like to thank my collaborators Dr Eunice Santos, Dr Eugene Santos Jr., Dr. Hien Nguyen, Dr. Long Pan, Dr. Qunhua Zhao, Huadong Xia, Fei Yu and Deqing Li who worked with me on the research project leading to this work.

My parents have been instrumental for any success I have had in life. The early and constant encouragement of my parents helped to instill in me a love for science and persuaded me to go for graduate studies. To them I dedicate this work. I would like to thank my brother and his wife for their support. My friends Ramesh, Kamesh, Mahesh, Vishnu, Karthik, Srirang, Deepti, Anand, Rajesh, Soshant and Krishna have made these long years at Virginia Tech memorable. Last but not least, I thank my wife Riya, who has been most patient these past few months and putting our lives on hold as I have worked to finish this dissertation.

This work has been supported in part by the National Geospatial Intelligence Agency.

Table of Contents

Table of Figures

Table of Tables

Chapter 1

Introduction

1.1 Modern Information Space - An Overview

The explosive growth of the internet [**1**] over the last two decades has been well-documented. For example, according to InternetWorldStats.com[1], as of June 2009, there are approximately 1.6 billion web users. An earlier study on the number of web pages and web servers had estimated it doubling every 18 months [**2**]. There is wide consensus that current tools for Information Retrieval (IR) are inadequate for dealing with the growing size and diversity of the internet. During the early years of the internet, the dominant form of internet content was text [**1**] [**3**]. Over the years, as the bandwidth available to users increased, other types of content such as images, large videos and audio files have been and continue to be uploaded in large numbers. It may be noted that typical multimedia files posted online are many orders of magnitude larger than regular text files. With the advent of new media such as blogs and social networking sites (e.g., Facebook[2] and Orkut[3]), ordinary users are uploading more of their personal information. In addition to the increasing diversity in content, there has been a trend towards hosting data that is updated or changed at a fast pace. Some online services such as financial sites, sports sites and online news sites update their content every few seconds. Overall, we are seeing the evolution of internet into a dynamic search space [**4**] with large and heterogeneous data. In fact, one of the most interesting developments in recent years has been the emergence of the "Real Time Web". The term is used to denote the real time data generated by applications such as Facebook and Twitter[4]. Twitter, a micro-blogging application, allows users to send out short messages (referred to as "tweets"). Tweet traffic has increased multifold in a short span of time, with users sending cumulatively millions of tweets a day. Conventional search solutions will not work with such rapidly changing information spaces.

[1] http://www.internetworldstats.com/emarketing.htm
[2] http://www.facebook.com
[3] http://www.orkut.com
[4] http://twitter.com/

Classic IR methodologies and technologies were primarily developed for static search spaces. Unfortunately, these cannot be directly applied to large real time data. A typical search engine, such as Google[5], works by scouring the internet with software agents called “crawlers” [**5**], extracting the information in the documents and creating indexes of the information [**6**]. Depending on the particular search engine technology, the indexes may contain keywords, positional information and PageRank [**7**] of the document. When a user queries the search engine, these indexes are used to determine the relevant documents. Since the internet has copious amounts of data, indexing is a computationally intensive task and takes a long time to complete. Hence indexing can only be done infrequently (e.g. monthly). Obviously such methodologies cannot be used with the data generated within the real time web. In fact, Larry Page, co-founder of the search giant Google has admitted that "I have always thought we needed to index the web every second to allow real time search" [**8**]. Bing[6] and Google search engines have taken the first step towards this by providing real time search of messages on Twitter and Facebook.

A prime example of an application, where real time retrieval of dynamic data is critical, is online news search application [**9**]. It has to pick the most relevant news articles from an increasing number of online news websites. In such cases, the classic indexing methods used by search engines do not work. News sites are popping up at a fast rate all over the world, and an increasing proportion of these websites provide minute to minute updates. Such sites should be indexed more frequently than websites with static content. This will lead to a strain on the computing resources available for the search application, resulting in performance degradation. As such, there is a great need for new and novel approaches and methodologies which can effectively deal with real time search on dynamic data.

One of the important reasons that current methods cannot fully adapt to real time retrieval in large and dynamic spaces is its reliance on offline indexing. The documents are processed and the extracted information content is stored in data structures such as inverted indexes [**10**]. The disadvantage of such

[5] http://www.google.com
[6] http://www.bing.com

offline document processing is that the actual querying process can only begin after all the processing/indexing has been done. It is clear that such a methodology cannot work efficiently in real time domains where documents enter and leave at arbitrary times. Also, a real time user cannot wait for the new documents to be processed before querying as this could take hours. Although modifications of classical methods have been proposed to deal with changing search space, they are simple alterations based on updating indexes [**11**] [**12**] at regular periods of time. Using current processing techniques with dynamic data will create strain on the computational resource and is not scalable in the long run. Clearly, the allocation of resources should quickly respond to changes in the search space. In order to design such dynamic allocation algorithms, dynamism in the search space should be explicitly represented in the retrieval methods. The processing methodology should be augmented by an intelligent document selection algorithm that provides prioritized resource allocation.

As such, we propose to use partial processing that uses anytime algorithms [**13**] [**14**] to process the documents. Anytime algorithms are interruptible and provide solutions whose quality improve with increase in processing time. Under the partial processing methodology, a document is processed in multiple steps. The duration of each step can vary according to the amount of resources allocated. After each step, a partial similarity value of the portion that was processed is calculated. The partial similarity value of a document provides an indication of its potential to be relevant and becomes the basis for its selection for further processing. As we can see, this leads to a prioritized selection of documents for processing and there is a better chance that relevant document will get processed faster than in traditional methods. Also, users don't have to wait for all documents to be processed but quickly get preliminary results based on the partial similarity values. These partial results get refined gradually and the users get better search results as the processing continues.

Data heterogeneity is another important issue in the modern information space. Documents on the internet contain not only text but may include images and streaming videos. Internet also brings together private

databases such as digital libraries for IEEE[7] and ACM[8] that hold the various publications of the two organizations. Hence heterogeneity is not only due to multiple data formats but also due to the difference in the internal schema of databases. Dealing with heterogeneity in schema may be solved by using metadata information of the individual schema [**15**] [**16**] for modifying queries before being sent to the different databases. Without a doubt, retrieving documents that are of different types (also called multimodal information retrieval [**17**]) and providing unified ranking is much more challenging. The straightforward method is to use distinct search methods for each data type and then integrate the results based on their similarity. However, an image with the same similarity value as a text document may not be equally relevant. Hence, some form of scaling [**18**] of the similarity values is required. Also, there are mixed documents such as web pages containing both text and images. Combining the similarity measures from text and image retrieval on such mixed or multimodal documents is also non-trivial. Using weights to combine their values is difficult as the problem of assigning appropriate weights is not well understood. A more intuitively sound methodology is to represent the information in heterogeneous documents using a common knowledge representation. By mapping information from different data to a common representation, the similarity values of documents from different sources or even different formats can be compared in a "fair" manner. Recent retrieval technologies have moved from using syntactic features such as term frequency, document frequency, etc. to understanding the meaning or semantics of the documents using knowledge representation such as concept graphs [**19**] [**20**]. We propose to focus on using semantic retrieval methods within the partial processing paradigm to solve the problem of heterogeneity.

Although partial processing allows for fine grained dynamic resource allocation, the issue of designing efficient resource allocation algorithms remains a challenge. Developing scalable resource allocation methodologies in large and dynamic search spaces has well studied issues. Resource bound decision making and partial observability [**21**] of large state spaces (for decision making) are three of the main

[7] http://ieeexplore.ieee.org/
[8] http://portal.acm.org/

challenges. We present a multiagent framework for designing and performance analysis of resource allocation algorithms that will work with approximate values produced by partial processing. Since the sequence of partial values generated by the anytime algorithm is non-decreasing, the changes in the search space are more structured. These structured changes are represented in a model which uses a grid-based representation of the search space. The search space provides a concise representation of the large state space and models the changes in document values and resources as flow of documents and agents between grids. The search space model is the lynchpin that interlinks the remaining components of our resource allocation framework. The framework is modular with each component dealing with different aspects of multiagent resource allocation such as agent communication, prediction and document selection. The partial processing paradigm along with the resource allocation framework provides a comprehensive solution to real time information retrieval in large and dynamic search space.

1.2 Scope of Research

It is clear that with the proliferation of real time information, the twin issues of size and dynamism must be addressed for the development of efficient retrieval methodologies. We will show how the challenges can be met by partial processing, and how partial similarity values can be used to produce quick search results with the help of adaptive resource allocation strategies. We will discuss how it enables us to provide focused resource allocation that zero in on relevant documents more quickly than other methodologies. Within the information retrieval field, there are a number of interesting research problems related to real time data such as information filtering [**22**], relevance feedback [**23**], etc. We will limit our discussion to the information retrieval problem of ranking documents based on their relevancy. Relevancy ranking of real time data is a harder problem than filtering where documents are simply labeled as relevant or not. As such, relevancy ranking is a crucial goal in the field of information retrieval that will lead to significant improvement in the quality of experience for the users. Since most of the data on the internet and real time databases are unstructured data, we will concentrate on methodologies using natural language queries with unstructured heterogeneous data.

1.3 Our Approach

The offline index processing which forms the backbone of current retrieval technologies requires large computational resources to keep up with the growing size of the internet. However, real time data which is updated every few seconds has proven to be a tougher problem, which cannot be solved by simply adding more computational resources. On the other hand an intelligent allocation of resources would be better suited in dealing with the scalability issues of large and dynamic data. In intelligent resource allocation, relevant documents are given priority over non-relevant ones. However, documents are known to be relevant only after being processed. How can this seemingly difficult issue be resolved? This issue is handled in the partial processing methodology by basing resource allocation decisions, such as position of a document in the processing queue and the amount of resources allotted to a particular document, on the partial similarity values generated in each step. Thus documents that show promise are able to quickly access computational resources and this will reduce the time to identify the relevant documents. The partial similarity values also enable us to provide a rolling display of the top relevant documents. This is essential in a changing search space where documents are continuously entering and leaving the databases. Similarity approximations in partial processing is made possible by development of anytime algorithm that are able to use the previous partial similarity values along with the information gleaned from the documents in the current processing cycle to produce the updated similarity values. This refinement process of the similarity value is not feasible unless some of the challenges are met. They are 1) anytime algorithm should be able to interpret information using only a part of the document, 2) the refinement process should converge to the correct final similarity, and 3) overhead due to multiple interruptions in processing should be minimal. We will further discuss these issues in this chapter. Since we use natural language queries with non-text documents such as images, we convert the information in the documents into a common knowledge representation called concept graph. By converting information from all documents irrespective of format into a common representation, we can directly compare their similarity values and provide a unified ranking system for heterogeneous databases. We provide validation of partial processing through simulation experiments and statistical analysis. The initial

validation has already been done within a multiagent IR framework called Intelligent Foraging, Gathering and Matching (I-FGM) [**24**] [**25**] using text documents. We provide further validation of this idea by designing image retrieval algorithm for images [**26**] [**27**] [**28**] within I-FGM and demonstrate its efficiency by comparison with current image retrieval methodologies. Image retrieval is a much harder problem than text retrieval because semantic gap [**29**] issues are acute with this data type. By validating partial processing with images, we can demonstrate its wider applicability.

Another important question that we tackle in this dissertation is how to model the resource allocation process that uses partial similarity values. A wide range of allocation strategies can be adopted. Simple allocation schemes may consider a document's partial-value in isolation while more complicated ones will compare it with other documents in the search space. We may also have to take other questions into account. Should the allocation have bias towards new document? Should documents with high partial values be given priority every time? The dynamism in the search space should also play a role in deciding how the resources are allocated. A resource allocation framework that can act as a foundation on which efficient resource allocation algorithms can be designed is required. We present a multiagent based resource allocation framework that models the unique characteristics of a large and dynamic search space where partial processing is employed. The framework not only helps in design but also in performance analysis of the allocation strategies. The framework is validated by designing and comparing allocation strategies on it. We provide a detailed statistical analysis of effects of the various factors on real time retrieval. In the following sub-sections, we briefly introduce the I-FGM framework followed by the challenges of real time search in large and dynamic search space. We will then discuss some of the research questions that we tackle in this dissertation.

1.3.1 I-FGM Framework

Intelligent Foraging, Gathering and Matching (I-FGM) framework [**24**] [**25**] [**26**] is an established real time information gathering framework that uses the partial processing paradigm. It has a multiagent architecture which consists of autonomous agents capable of interacting with each other to work in

complex, uncertain environment towards a common goal. The agents have specialized functions and their response changes with the interaction with the environment. Its modular nature allows for quick incorporation of new retrieval technologies without system wide changes, helping to deal with evolving technology. It provides a platform on which real time retrieval methodologies for different data types may be designed and tested.

1.3.2 Challenges

For real time retrieval in large and dynamic search space, resource allocation within a distributed processing environment is required for providing solutions in a time bounded manner. We elaborate on the issues that have to be addressed for formulating a methodology for real time search.

Online Processing

Although the real time characteristic of the data has made the search space dynamic, the dynamism has not been clearly understood or modeled. How does it affect retrieval efficiency? Do we need new definition for metrics such as similarity and recall? The main thrust of research has been in modifying existing retrieval methodology such as inverted indexes, for the changing search space [**30**] [**31**]. It is estimated that search engines will need 1.5 million computing nodes to keep up with the internet content and the querying rate in 2010 [**32**]. Since this is not scalable, a new way of looking at the problem is necessary. The traditional methods are essentially offline processing paradigms which construct indexes out of all the information in the search space. However, when dealing with large volumes of real time information, it is essential to have online processing, which can deal with new incoming documents or updates in existing ones. Then the critical questions that online processing algorithms have to answer are: 1) which documents to process? 2) when to process them?

Architecture

The challenges posed by the computational requirements of large and dynamic spaces have led to gradual migration from client-server model of web search engines to distributed architectures such as multiagent

[**33**] and Peer-to-Peer [**34**] . The search engines are based on the paradigm of centralized indexing where the documents are pulled into a central repository and processed. Due to the decentralized control in distributed architectures, the processing of the data is tightly coupled to the needs of the user or the prevailing conditions in the search space. They have the autonomy and flexibility to react quickly to the changing resource requirements in large and dynamic search spaces. Although architecture issues such as agent modeling, distributed file system and network routing issues have been studied in detail for multiagent and P2P architectures, they were done with respect to particular application domains. The distributed processing research approaches within the IR domain have taken a laissez-faire approach towards dynamic search spaces and have not explicitly modeled the dynamism. In this dissertation, we will show that a dynamic search space model is crucial towards formulating an effective framework for real time search.

Resource Allocation

One of the issues with distributed architecture with no centralized control is that the individual nodes or agents have to make decision with only local information about the search space. Decentralized architecture is essential if scalability has to be maintained with respect to limited computational resources and increasing search space size. The issue of dynamism also makes the resource allocation more complicated. A methodology for formulating resource allocation policies and determining whether they are effective is one of the critical issues that have to be studied.

Emphasizing Real time in Algorithm Design

Dynamism brings to the forefront the notion of prioritizing resource allocations. Clearly, most recent documents need to be given more importance. Current algorithms do not take this into account. The relevance of the real time information also changes with time. How do we calculate the relevancy by incorporating this information? Real time information may also come with deadlines after which the information becomes stale. Processing stale information is a waste of time and resources. These aspects have to be incorporated while designing algorithms for real time retrieval.

Multiple Retrieval Methodologies

Since heterogeneous search spaces contain text, images and multimedia files, multiple retrieval algorithms have to be applied, each working on a specific type of document. There are also multimodal documents that contain multiple data types. Prime examples of multimodal documents [**17**] are web pages containing embedded images. Even in homogeneous search spaces, multiple retrieval methodologies may be used by the user. These methodologies may employ varied retrieval techniques such as vector models [**35**], probabilistic models [**36**] and language models [**37**]. The challenge here is to make these techniques to work together. How does a document *a* ranked *x* by method *l* compare to a document *b* ranked *y* by method *m*? Although Metasearch [**38**] and Federated Search [**39**] deal with heterogeneity, they are limited to merging results brought in by different search engines from databases with different schemas. How do we construct a unified ranking that not only takes heterogeneity of data types into account but also the heterogeneity in retrieval technologies? This is a crucial question that needs to be answered to deal with the growing heterogeneous nature of the search space and the information needs of the user.

1.3.3 Research Questions

Applying partial processing to real time search brings to the forefront important research questions. Whether it is possible to interpret portions of the document and calculate the partial similarity measures without significant overhead is a question that needs to be answered before partial processing can be used as a solution. Since we are using the multiagent architecture, agent decision making using the incomplete information is also another challenge that needs to be addressed.

Research Question 1: Interpreting Partial Information

Even before the partial processing paradigm can be adopted, there is the question of whether retrieval algorithms can be designed that can work with only portions of the document at a time. How will the partial information generated in each step be interpreted? What are the issues that need to be considered while designing the retrieval algorithms that work with partial processing? Although partial processing

has been demonstrated with text documents [**24**] [**25**], it was only an initial validation. Automatic retrieval algorithms generally require the transformation of document data into data structures that can be used by software programs. For text, data structures such as inverted indexes [**5**] and latent semantic indexes [**40**] have been found to be very effective. Text retrieval tools have matured over decades of research, leading to adequate performance. For image retrieval, the images are converted into its low level features such as color histograms and wavelets [**41**]. However interpreting visual information using these low level features is extremely hard because human interpretation on image relies heavily on contextual information. This is also the case in other multimedia retrieval such as video. Therefore, it is to be expected that image retrieval algorithms that use partial information and have anytime qualities are going to be a challenge to design. To demonstrate the generic applicability of partial processing, we design a retrieval algorithm for a complicated document format such as images. We validate our image algorithm within the I-FGM framework and also show by experimental results how resource allocation strategy based on partial information can speed up the image retrieval process.

Research Question 2: Designing a Resource Allocation Framework

In traditional distributed IR systems, resource allocation consists of simple division of the documents among the computing nodes. The nodes process the documents and then merge the results. There is no notion of prioritized resource allocations as it is difficult to calculate the importance of documents even before its contents get processed. The intermediate relevancy values produced by the partial processing values provide a way to tentatively order the documents in the search space and prioritize the resource allocation. This ordering changes as more of the documents are processed and its relevancies refined. Other resource allocation frameworks [**42**] [**43**] do not deal with the specific issues of dynamism brought on by partial processing. We believe that partial processing using anytime algorithms have special characteristics that can be leveraged by existing methodologies for better performance. Towards this end, we present a multiagent based resource allocation framework that has a search space model that takes into account various types of dynamism such as change in document ordering, change in agent behavior and

mobility. Its modular nature allows easy plug-n-play of various algorithms into a particular component without changing the rest of the components. The main components of the framework are the agent model, interaction module and the search space model. We show how the components interact with each other to model different processes in the search space. We use a detailed statistical analysis of the experimental results to validate the framework.

1.4 Overview of the Dissertation

We give a brief survey of the current information extraction and processing technologies, with a focus on web documents in Chapter 2. The research described in this dissertation grew out of the work on developing the I-FGM framework. As part of the background material in Chapter 2, I-FGM is described in detail. An anytime text retrieval algorithm using concept graphs which was used to validate the I-FGM framework is also described. Text retrieval is also the preliminary validation of partial processing and demonstrates how partial information can be used to calculate similarity approximations. The main results are described in chapters 3 to 6. Chapter 3 describes the details of an anytime image algorithm that was designed to work within the partial processing paradigm. The various components of the algorithm along with the metrics used to guide resource allocation is also described. The validation results of the image algorithm are given in Chapter 4. The resource allocation strategies employed with the text and image algorithms were restrictive and did not model the dynamism of the documents in the search space. In Chapter 5, we formulate a resource allocation framework for multiagent system that takes into account the various issues involved with large and dynamic spaces. We describe the various components in the framework and how they interact with each other. In Chapter 6, we implement a resource allocation strategy based on our framework and by comparing its performance with control systems, we validate our framework. We also provide a statistical analysis of the effect of various factors on system performance. The figures and results presented in the dissertation have been published in [**24**] [**25**] [**26**] [**27**] [**28**] [**44**] [**45**].

Chapter 2

Background and Literature Survey

2.1 Introduction

In Chapter 1, partial processing was briefly introduced as a way to solve the twin problems of size and dynamism for real time retrieval. We continue our discussion on partial processing in this chapter by describing it in detail. In order to truly understand the significance of partial processing, it is important to discuss current retrieval methodologies. We focus on web retrieval technologies as it is one of the areas where a real time search solution will have a major impact. In recent years, distributed IR has been proposed as a solution to deal with large and dynamic search. We will survey some of these distributed architectures and point out their strengths and weaknesses. As part of the background for partial processing, we describe the Intelligent Foraging, Gathering and Matching (I-FGM) framework. I-FGM is a multiagent IR framework that introduced partial processing as a way to do dynamic resource allocation in large and dynamic search spaces. As part of the work in I-FGM, an anytime text retrieval algorithm was developed that processed text documents incrementally. Appropriate metrics and algorithms that use similarity approximations to guide document selection and resource allocation were also developed. We review the text retrieval algorithm that have been published in [**24**] [**25**] [**44**], stressing the issues involved in design and implementation.

2.2 Terminology

Before going into the details of partial processing, we define some IR terms that are frequently used in this discussion.

1. **Document**: is the basic unit of retrieval. In [**35**], Baeza-Yates & Ribeiro-Neto defines document as "a unit of retrieval. It might be a paragraph, a section, a chapter, a Web page, an article, or a whole

book." In this dissertation, the term document refers to text or image file that is being measured for relevancy to a particular query.

2. **Search Space**: is a collection of documents that are being accessed by the user. It is also referred to as "information space" or "database" in this dissertation. The search space may be located in one physical location or distributed among multiple locations.
3. **Query**: is defined as "the expression of the user information need" [**35**]. We use a complete natural language sentence as query in the retrieval algorithms discussed here.
4. **Similarity**: Similarity of a document is measured with respect to a query and is the quantitative measure of how completely the document satisfies the information needs of the user as encoded in the query. The actual methodology for calculating the similarity varies with the algorithm, application or information need. Similarity is also referred to as relevancy and is a real number in [0, 1.0].
5. **Recall**: is defined as "performance measure that quantifies the fraction of known relevant documents which were effectively retrieved" [**35**]. It is a real number in [0, 1.0].
6. **Precision**: is defined as "performance measure that quantifies the fraction of retrieved documents which are known to be relevant" [**35**]. Recall and precision are commonly used IR metrics.

2.3 Related Work

Since the internet is a prime example of a modern information space with large and dynamic data, we will first survey some of the web retrieval technologies. Indexing is the dominant document processing technology used by web search engines and we provide a summary of the various improvements in indexing. However, the success of indexing based real time methods depends on being able to process and create indexes of the documents in the search space as quickly as possible. With the internet growing at unprecedented rates and with real time information being made available, indexing technologies must depend on large computing resources in the form of server farms to provide quick results. Clearly, this is not sustainable. We survey alternate distributed architectures that show great promise for its ability to deal

with dynamism better. Further, we describe a promising set of algorithms called anytime algorithms that are designed to work in resource bounded situations.

2.3.1 Web Retrieval

Indexing [**5**] [**35**] is the dominant information extraction and storage methodology for extracting and collating information from web documents. There are many types of indexes such as inverted index [**10**], signature file [**46**] and b-tree index [**47**]. Inverted indexes, being the most efficient [**10**] [**48**], is the most commonly used type. Although different search solutions have indexes that differ in their constituents, its basic structure consists of vocabulary list and posting lists. The vocabulary list consists of all terms from all documents in the search space. Each term has a posting list which contains the document IDs of the documents containing the term, its frequency and location in the document. Other contextual information such as link structure (for the PageRank algorithm [**7**]) may also be present. Web search engines employ crawlers [**5**] which are software programs that explore the internet and index the web pages. Since the concurrency of the indexes depend on the periodicity of the crawling, the more dynamic the web pages, the more frequent the indexing should be. On the other hand, indexing is a computationally intensive task [**11**] and could take weeks to complete [**49**] in large search spaces. Many search engines sidestep this issue by considering the internet to be fairly static and only conduct periodic indexing. Therefore, information entering a particular website between two indexing cycles is essentially unavailable to the user. Newer indexing algorithms do consider dynamic data. This is especially difficult in large search spaces as they would also have large indexes. Updating such indexes frequently has adverse consequences on performance. For computational efficiency, the updating methods rely on hardware-centric techniques such as maximizing in-memory computation, contiguous storage on disk and block I/O operations [**48**].

Index update algorithms may be broadly classified into three categories [**10**]:

1. **Rebuild**: Rather than updating the current indexes, they are rebuilt from scratch. This is a straightforward albeit computationally inefficient method and is only feasible for small databases.
2. **Intermittent Merging**: The main index is stored on the disk and a smaller auxiliary index, which is updated with information from the new documents, is kept in-memory. When the in-memory index runs out of space, it is merged with the disk based main index. For a given query, the search results from the main list and the auxiliary list are merged and relayed to the user.
3. **Incremental Update**: A drawback with intermittent merging is that, at the time of merging of the in-memory based auxiliary and disk based main indexes, the list will not be updated. Incremental update is similar to merge except that the main list is updated term by term. This allows for a low priority background process to do the update of the main index so as not to interfere with query processing.

Although some of these techniques mitigate the effects of expensive index update operations, they are not scalable in terms of managing large number of queries or index sizes [**50**]. Other methods focus on modeling the effects of dynamism. Incremental crawling [**49**] and Just-In-Time-Indexing [**11**] are two such techniques. In incremental crawling [**49**], the frequency with which the crawlers visit web pages depend on the perceived rate of change of its content. The changes in the pages are monitored and the crawling frequency is modified accordingly. Just-In-Time-Indexing [**11**] was proposed to index parts of the main index using information that arrived between two cycles of batch indexing, so as to provide a real time search capability. But this solution was designed for medium-size (based on query frequency) search systems such as intranet searches. **S**ub-document **i**nde**X**er (SiX) [**12**] is a specialized inverted index design that supports updates to sections of a document without having to re-index the whole document.

Typically for large databases, the index generated is also large and is spread over multiple computing nodes. Hence distributed indexing algorithms are widely used for large search spaces. Distributed indexing may be divided into two main types: global indexing and local indexing. The difference lies in the way they store the indexes. In global indexing, the index is distributed among the nodes in the cluster

and each node is assigned a set of terms. After a node processes the documents, it distributes the index terms to appropriate nodes. In local indexing, the node carries index terms for only the documents assigned to it. When a query is processed with global index, it is sent to only those nodes that contain the terms corresponding to the query. When a query is processed with local index, each node sends a set of relevant document IDs and there is a merging of these results to produce the final relevancy list. A study in 2002 [**51**] has found that global indexing has better performance than local indexing. Among the many distributed indexing solutions [**52**] [**53**] available, MapReduce [**54**] is an industry standard distributed processing framework developed at Google. It is designed to work with large clusters made with Commercial-Off-The-Shelf (COTS) components. MapReduce divides the document collection into smaller collections or splits and assigns them to individual machines. The indexes generated by individual machines are merged. Index values are sorted according to the keyword terms and divided among the machines. The framework employs a master node for task division, resource allocation and maintaining fault tolerance.

2.3.2 Distributed Information Retrieval

In the previous section, we looked at indexing and how distributed indexing is done to handle large amounts of data. In the applications described above, the web crawlers brought in the information from many websites and used the hyperlinks to navigate from one web server to another. There are instances where the information required is in private, restricted databases. In fact, there has been a proliferation of such types of databases on the internet and this section of the internet has been termed as the "Deep Web" or the "Invisible Web" as opposed to the directly searchable web or surface web [**55**]. Examples of hidden websites are government websites such as the US Census Bureau[9] and subscription database like LexisNexis[10]. They are large databases that have their own search engines. In this section we describe distributed IR which deals with accessing multiple databases or search solution at the same time. Based on the application domain, distributed IR may be termed as metasearch [**38**] [**56**] [**57**] or federated search

[9] http://www.census.gov/
[10] http://www.lexisnexis.com/

[**39**] [**58**]. Metasearch is frequently used to imply web retrieval whereas federated search is used for searching private database. Metasearch engines send queries to multiple web search engines, merge its results and display the concatenated results to the user. Commercial metasearch engines such as Dogpile[11] and Clusty[12] perform simplistic merging and categorization of the results. One of the aims of distributed information retrieval is to provide the user with a unified search interface that can search multiple, physically distributed databases by using its propriety search engines or utilizing the database schemas. The main research issues [**59**] are: (1) modeling individual databases including its data model and information characteristics [**58**] [**60**], (2) selecting appropriate search engines and data sources [**61**] [**62**], and (3) merging the results from multiple sources [**39**] into a unified ranking system

The traditional web search engine has a client-server architecture where queries are received from the web browser or the client. The server is the search engine which fetches the relevant results by using the indexed information in its central repository. In federated search [**58**], each document source is serviced by a document server which is like a search engine and only processes the documents under its control. The client sends a query to the server broker which forwards it to a select subset of document servers. It models the performance of documents servers and makes the selection based on their perceived utility. The brokers merge the results from the document servers before sending them to the client.

Even the traditional web search engine and the federated search do not scale well for large databases. The web search engine employs crawlers to pull documents to a common location before processing them. The processing and storage of documents in the repository could lead to a bottleneck. In federated search, the broker is the bottleneck [**63**]. Alternate decentralized, distributed architectures in the form of peer-to-peer [**34**] and multiagent [**64**] [**33**] help to overcome this weakness.

In the peer-to-peer (or P2P) [**34**] architecture, each node acts both as client and server. Each node seamlessly shares its resources, such as storage and processing, with other nodes or peers through an

[11] http://www.dogpile.com
[12] http://www.clusty.com

overlay network. An overlay network [**65**] is the communication topology of the network and is crucial for routing messages. The advantage of P2P architecture is absence of a centralized control which enables asynchronous operations. P2P is also highly scalable with the amount of resources deployed limited only by the number of participating nodes. P2P applications such as Napster and Gnutella [**66**] have been very successful in handling file sharing [**67**] [**68**] among large number of subscribers without a centralized storage for the files. In information retrieval, P2P systems offer an alternative to traditional web retrieval/indexing architecture by its ability to work on highly distributed data in-situ and by its scalability. P2P systems can be broadly classified into structured [**69**] [**70**] and unstructured [**71**] based on the overlay network. In structured systems, each node is given a unique ID which is published in a Distributed Hash Table (DHT) [**69**]. The DHT is shared among many nodes with in-built redundancy to reduce disruptions by node failures. Each document is also given a unique ID on the DHT which is used to route messages or retrieve a document. Hence the communication is more directed. In an unstructured network, no such directory service exists and therefore routing is conducted by flooding protocols. For example, Napster had a structured network wherein the file locations were listed in a central repository. This made it vulnerable to attacks and lawsuits. Gnutella, which is similar to Napster, did away with the centralized system. For highly redundant data, the unstructured network gives a good performance while the structured network works well even for rare documents. But constant retrieval of the popular documents will decrease the performance of structured networks. Several IR solutions have been proposed with P2P networks. A P2P architecture for IR has been described in [**72**] by abstracting the essential functionalities into separate layers. One of the issues with using P2P with large search space is that the publishing of large indexes is not scalable. Some solutions based on publishing only important or widely used keywords have been proposed. A variant of the P2P architecture called the hybrid P2P [**73**] [**74**], which has super nodes that will publish the information of the documents in the nodes under its authority have been proposed to mitigate this scalability issue. Apart from the ability to scale to large number of nodes and large databases, P2P IR architecture is also highly fault tolerant and allows for flexible deployment of resources. The downside is that routing of messages over the overlay network [**65**]

becomes crucial for achieving good retrieval performance and this could be a challenge in widely distributed P2P systems.

A multiagent system may be defined as a set of autonomous, goal driven, interacting agents that work towards a common purpose. They have similarities to P2P system such as decentralized control and asynchronous operation. P2P is more concerned with message routing and resource sharing whereas multiagent systems focus on modeling agent functionality, behavior and interaction. The Harvest [**75**] project provided an early multiagent distributed architecture for accessing information available on the internet. It has component based architecture with separate components for searching, downloading and processing data. It stores the information of downloaded documents in the form of indexes which are then used to retrieve relevant documents for a given query. InfoSleuth [**76**] is another agent based IR system that does searches in heterogeneous databases. It uses ontology to construct the semantic relationships between the fields in the databases. It has agents with different functions that include accepting queries, selecting suitable databases, converting the query into a form accepted by the databases and sending the queries to the databases. But it is only limited to retrieval from relational database. ACQUIRE [**77**] is a mobile agent [**78**] system that performs heterogeneous data retrieval. Mobile agents are software programs that travel the network to locations where data is located to perform a particular task as opposed to getting the data from the database and processing it at the place of program initiation. ACQUIRE is an on-demand type of system in which mobile agents are only activated when a query is received by the user. The mobile agents then moves to the various databases under its purview and searches for relevant data.

2.3.3 Anytime Algorithms

The methods discussed above deal with large databases by developing innovative parallel and distributed architectures. The notion that information retrieval has become a resource bounded (both time and compute power) activity remains generally absent from these solutions. There is a need for retrieval algorithm that explicitly models resource/time trade-offs in its design. One such area in Artificial

Intelligence that looks at resource bounded problems is anytime algorithms [**13**] [**14**]. They are interruptible algorithms that produce solutions whose quality is proportional to the amount of computational resources applied. Anytime algorithms are defined as "algorithms that return some answer for any allocation of computation time, and are expected to return better answers when given more time" [**13**]. They have been successfully applied in real time problems such as robot navigation [**13**], scheduling [**79**] and real time search algorithms [**80**]. Anytime algorithms have certain desirable characteristics that set them apart from other resource bounded approximation techniques. They are [**81**]:

1. **Interruptibility:** The algorithm may be stopped at any point and some approximation of the final result is delivered.
2. **Preemptability:** The property of anytime algorithm by which it may be restarted with minimal overhead.
3. **Result Quality:** It is possible to quantify the quality of the result. The quality increases with the application of more computational resources.
4. **Predictability:** It is possible to predict, with some accuracy, the result quality for the amount of resources applied.

Although anytime algorithms have not been widely used in information retrieval, we will survey some work in related fields such as resource bounded information gathering. Information gathering [**82**] [**83**] is an activity designed to support human decision making process by collecting relevant information from sources such as the internet. Since the decisions have to be done within resource limitations such as bounded network bandwidth or within time limits, anytime algorithms are used to model the time/result quality tradeoff. They are used to calculate the appropriate number of queries and select sources so as to ensure a particular quality of results. Image alignment [**84**] is a problem in computer vision which deals with measuring the correlation between two images of a scene taken at different time steps. This correlation is measured in real time and anytime algorithms are used to calculate the size of the pixel

sample used for comparison so as to satisfy the time constraints. The bigger the sample size, the better the result quality but longer the computation time.

2.3.4 Summary

Distributed systems such as multiagent systems allow for quick and flexible deployments of resources. The decentralized architecture in these systems makes them scalable and helps to avoid the bottlenecks found in traditional client-server architecture. Although there are many distributed IR architectures that deal with large search spaces, they do not perform intelligent resource allocation. All documents are considered to be equally likely to be relevant and there is no notion of priority in their processing schedule. Anytime algorithms provide a framework to quickly find approximate similarity values that can be used to perform a coarse grained categorization of documents. This categorization helps in a focused and intelligent application of computational resources. The concepts of similarity approximation and document categorization are the main ideas behind the partial processing methodology.

2.4 Partial Processing Paradigm

The current methods of processing information are geared towards static information spaces. The dominant technology of offline indexing of the databases is computationally expensive and is not scalable. The stopgap methods adopted in the form of large scale cluster/distributed computing cannot keep up with the growth in the size of search spaces. It may be argued that since the computing power is growing at faster rate (according to Moore's law [**85**]) than the internet [**86**], a brute force methodology of using larger computing clusters is a viable solution. Ultimately, this becomes a hollow argument when we take the dynamism of the data into account. When a document is updated, it has to be reprocessed by the current retrieval algorithm. So although the document update does not cause an increase in the search space size, computational resource is nevertheless expended. Hence the notion of equating computing power to the size of the search space for calculating resource requirements is misleading.

Although new architectures based on large scale parallel/distributed are essential for rapid query processing, the performance bottleneck created at the document level has not been adequately addressed. In order to mitigate the effects of this bottleneck, a novel information extraction and processing paradigm called partial processing that incrementally processes a document and intermittently calculates a partial similarity value using the information processed until that point was proposed in [**24**] [**25**].

We illustrate the partial processing paradigm in Figure 2-1 where a queue of documents is processed with the traditional full processing and partial processing paradigm. We see that in partial processing the documents are broken into pieces and the order in which the documents are processed depends on the partial relevancy value. In partial processing, the wait time for relevant documents can be reduced whereas in traditional processing, it is rigid. Hence there is a possibility that the relevant document is processed faster with partial processing. This depends on how well we predict the future similarity value of the documents using its partial-values.

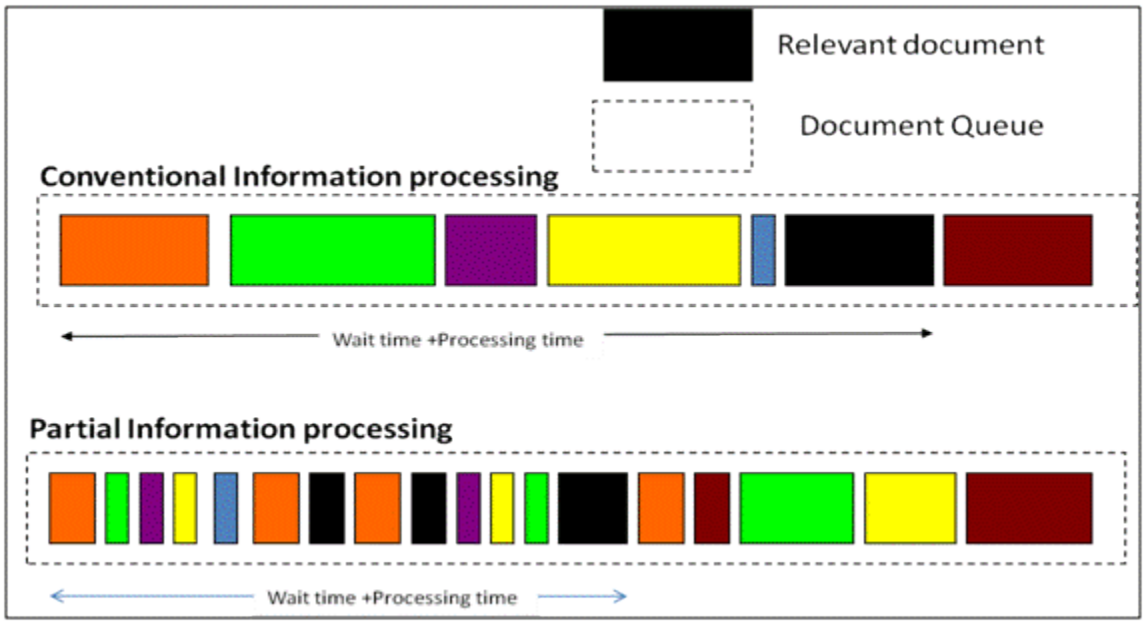

Figure 2-1 Partial processing paradigm

The main points of difference between the traditional or full information processing and the partial processing are:

1. **Similarity Calculation**

   a. Traditional: Documents are processed in a monolithic fashion and similarity is calculated after processing it entirely.

   b. Partial: Documents are processed incrementally and the similarities are calculated from the information processed till then. Algorithms with anytime property are employed to extract information from the documents. Anytime algorithms provide solution quality commensurate with the amount of computational resources applied. The quality of solutions increase with increase in resources applied to the algorithm.

2. **Resource Allocation**

   a. Traditional: Resource allocation is coarse grained and is focused on distributing the documents or workloads among the participating nodes.

   b. Partial: Resource allocation at individual document level is possible. Each incremental processing interval allotted to a document can be varied according to its potential to be relevant. This type of fine grained resource allocation can reduce wastage of resources by trying to prioritize allocations to relevant documents based on its partial similarity. Preventing wastage on non-relevant documents is important because for a typical query, relevant keywords occur in only 0.1%-1% [**10**] of the documents in the search space.

3. **Query Results**

   a. Traditional: Documents are ranked based on the similarity of completely processed documents. But in dynamic search spaces, with new incoming documents, the target set of relevant document changes with time. Hence the result display should also be dynamic and changing.

   b. Partial: Interim results are calculated using the partial relevancy values and there is a refinement or convergence towards the actual set of relevant documents as the processing progresses. Thus the dynamic nature of the search space is taken into account. And the values of the new and updated documents are quickly incorporated into results calculation.

There are some issues related to the application of partial processing methodology to the IR field. These issues have to be addressed in the design of retrieval algorithms within the partial processing methodology.

1. **Prediction of future values**: The partial similarity values produced by the partial processing are used to guide the resource allocation process. Hence accurate predictions of future partial values are important for making decisions on the amount of resource to be allocated. Since there does not exist any reliable model on the distribution of relevant information across a document, making accurate predictions is a challenge. On the flip side, it may be noted that even in the worst case, partial processing paradigm will perform as good as the full processing paradigm. This is because the ranking and collecting of results in the full processing paradigm occurs only after all the documents in the search space are processed.
2. **Interpreting partial information**: The information extracted during the intermittent steps of the partial processing paradigm is used to calculate the partial similarity of the document. Although current information retrieval techniques do not explicitly use partial information, a cursory survey of lead us to believe that these methods can be modified to accommodate partial information. We will validate this notion in this dissertation by showing how similarity evaluation may be done with partial text and image information.
3. **Computational overhead**: The incremental processing and update of partial similarity values has an overhead that needs to be balanced. This overhead is directly proportional to the number of steps taken for a document to be processed. Hence selection of suitable methods for retrieval with partial information will depend on the overhead as it has an effect on how quickly the results are produced.

2.5 Text Retrieval with Partial Processing

Partial processing had been applied to text retrieval as part of earlier work [**24**] [**25**] [**44**] for developing the I-FGM framework. An anytime text processing algorithm based on semantic retrieval principles was designed. Resource allocation strategies that used the partial similarity values were developed and its

performance analyzed. In this section, we describe this work including the design aspects of the text retrieval algorithm. For validation methodology and performance results, please refer to [**24**] [**25**] [**44**]. Before going into the details of the retrieval algorithm, we describe in detail the I-FGM framework.

2.5.1 Intelligent Foraging, Gathering and Matching (I-FGM) Framework

Intelligent Foraging Gathering and Matching (I-FGM) [**24**] [**25**] [**26**] [**28**] [**27**] [**44**] framework has the following goals: (1) retrieve relevant information from large and dynamic databases in a timely fashion, (2) facilitate efficient usage of computational resources and direct it towards potentially relevant documents, (3) unify the results from multiple retrieval methods, and (4) provide performance guarantees. I-FGM is a comprehensive real time retrieval solution that is based on multiagent architecture coupled with the partial processing paradigm.

An efficient real time system has to quickly retrieve relevant documents from a dynamic and vast search space. The traditional information processing paradigm that is commonly used by current retrieval methods is to fully process a document before it is evaluated for relevancy. In search spaces where the relevant set forms a small portion, this method of document processing leads to wastage of computational resources and large wait times. I-FGM introduced partial processing, in which a document is processed in multiple cycles, each cycle called a partial step. The part of the document processed in a partial step, called a nugget, is then evaluated for relevancy. This partial relevancy of the document gives an idea of the final relevancy of the document and is used to rank the documents in the search space. Based on the partial relevancy of a document, a decision whether to allocate more computational resources to that document is made. Thus, it is the search space that guides the allocation of the computational resources. Repeated processing of the documents refines its partial relevancy value until it reaches the final value. Consequently this refinement process continuously changes the set of relevant documents and these results are displayed in a rolling fashion. This is useful in dynamic databases where new documents stream in and relevant set of documents change.

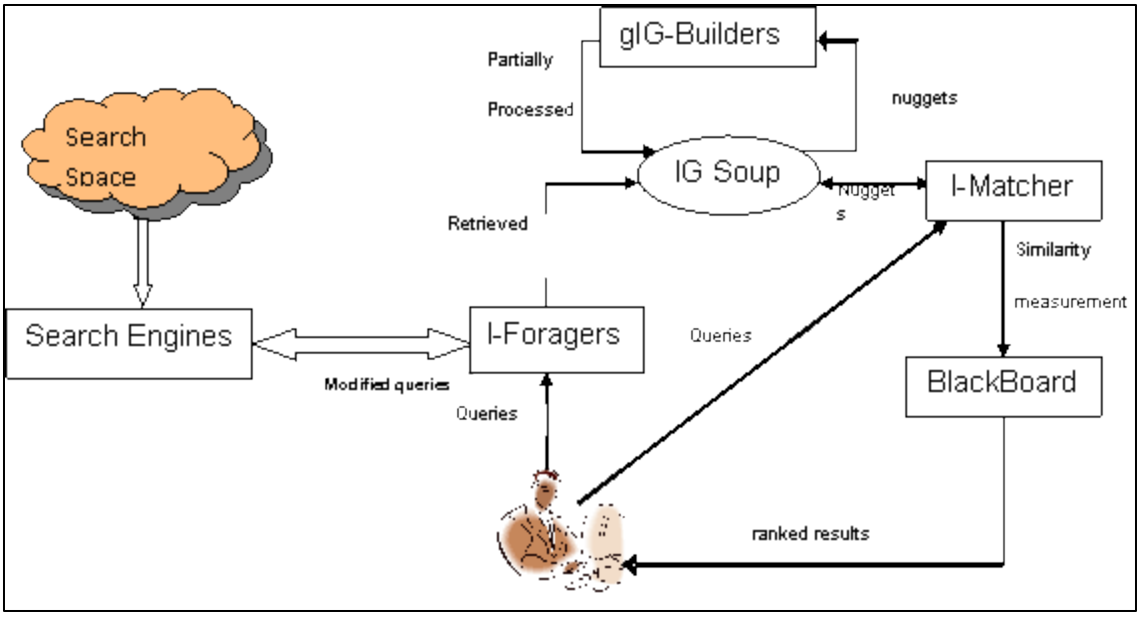

Figure 2-2 I-FGM framework

In order to implement this concept of incremental processing of documents, a flexible and dynamic architecture, capable of rapid deployment of computational resources is required. I-FGM provides exactly this. The fundamental unit of computational resource is regarded as a process. The architecture is dynamic in the sense that processes are able to quickly move from one function to another. The processes are grouped together according to its function to form components. The various components and its interactions are depicted in Figure 2-2. I-FGM searches (forages) for documents from various databases, downloads the documents into a local repository (gathering), incrementally process the documents and compares the processed documents against the query to determine (matching) the relevancy measure. I-FGM has several components to carry out these functions: 1) I-Foragers to gather and download data from various databases using third party retrieval methods, 2) gIG-Soup to store the downloaded and semi-processed documents, 3) gIG-Builders to process the documents, 4) I-Matchers to evaluate the relevancy of the processed documents, and 5) Blackboard to display the most relevant documents as results.

I-FGM uses a multiagent based architecture. Some of the advantages of using multiagent architecture in I-FGM are: 1) Flexible and modular architecture is possible, 2) It does not have centralized control, and 3) It allows for asynchronous computation. Since it is modular, third party retrieval tools can be easily incorporated into I-FGM as plug-and-play. Multiagent architecture is amenable to component based architecture and separate agents can be developed to do the foraging, gathering and matching, respectively. Asynchronous operation and absence of any centralized coordination help in rapid and efficient deployment of computing resources. In the remaining sections of the chapter, we discuss the components of I-FGM and results for text retrieval on I-FGM that were published in [**24**] [**25**] [**44**].

2.5.2 I-FGM Components

I-FGM architecture has the following components:

1. **I-Forager**: does a first-order search among multiple, heterogeneous databases looking for suitable candidates for download into I-FGM. I-Forager narrows down the search space for a particular query by using some third part retrieval tools. After it is downloaded, the documents are further processed and converted into a format/representation that can be analyzed for relevancy.
2. **gIG-Soup**: is the repository that holds the documents that are newly downloaded by the I-Foragers. Other agents select from among the documents in the soup for processing. Partially processed documents wait in the soup for additional resources.
3. **gIG-Builder**: are processes that scour the gIG-Soup looking for potential documents for further processing. The documents are converted into some common knowledge representation. By converting it into a common representation, we are able to adopt a common ranking across different data formats.
4. **I-Matcher**: selects documents newly processed by gIG-Builders and compare them against the query to get their updated partial relevance measures. The documents are then returned to the gIG-Soup for further processing.

5. **Blackboard**: displays the top documents to the user. The top documents are decided based on the current relevancy values of the documents in the gIG-Soup. As the refinement of the relevancy of the documents continues and as new documents arrive, the results appearing in the blackboard changes.
6. **Query Builder**: A natural language query is accepted by I-FGM. In order to compare the query with the documents in the gIG-Soup, the query is converted into the same representation as the processed data in gIG-Soup.

2.5.3 Text Retrieval - Design Considerations

In the following sections, we show how I-FGM was used to perform text retrieval. For any text retrieval algorithm to work with I-FGM, it must have anytime characteristics of interruptibility and predictability. It should also deliver a partial result at any intermediate processing step. A semantic methodology that had these properties was adopted for text retrieval. The content in the text documents is represented using graph structures called concept graphs [**19**] [**20**]. The algorithm processes one sentence at a time and adds the information from the newly processed sentence to the already constructed graph. Thus the algorithm can be interrupted at any intermediate point and still produce a graph consistent with the information extracted till that point. By comparing the concept graph of the document with that of the query, we can calculate the similarity measure without significant overhead. In the rest of this section, we provide background on concept graphs and describe the resource allocation metrics used.

Semantic Text Retrieval

The first generation of retrieval algorithms such as vector space model [**35**] was based on extracting and matching keywords. These methods have limitations as it does not involve understanding the semantic content of the documents. Hence the quality of results really depend on the wording of the query [**87**]. Also in standalone document without hyperlinks, link structures and consequently PageRank [**88**] cannot be used. A more effective content analysis approach is required. The current direction in IR is headed towards understanding the semantics of the documents and modeling the user needs through user modeling [**19**] [**20**]. The emergence of the semantic web [**89**], wherein regular web documents are

complemented with metadata [**90**] that help the retrieval algorithms in assigning meaning and context to the content, is another indicator of the importance of using semantic information in the retrieval process.

Concept graph [**91**] is a knowledge representation that models the semantic content as a set of concepts and relations. The nodes in the semantic network represent the concepts and edges connecting the nodes represent the relation between them. Although nodes may depict any concept, the set of relations used are defined beforehand. The relations generally used in these graphs may be categorized [**92**] as:

1. **Generalization**: is a relation connecting a general concept and a more specific example. E.g. the relation between a *car* and a *Ford car* is a generalization relation.
2. **Association**: is a relation connecting concepts that are similar in meaning. The instances of one such concept is also similar to the instances of the associated concept. E.g. the relation between a *Ford car* and *Toyota car*.
3. **Aggregation**: is a relation between an object and its constituent parts. E.g. *computer* forms an aggregation relation with its component *monitor*.
4. **Of-property:** is relation between an object and one of its property or feature. E.g. In the phrase *black cat*, *black* forms an of-property relation with *cat.*

We have used this knowledge representation in I-FGM as it can be used as a universal representation of heterogeneous information coming into I-FGM and provide a unified ranking system that is complemented by theory. It also makes compare and contrast of different types of documents possible. Also, we can cast the similarity calculation as a graph matching algorithm, providing a graph-theoretic basis to the similarity definition. Besides, concept graphs have become popular due to the ease of incorporating reasoning techniques [**93**] for analyzing the information. We will refer to the concept graph generated from the contents of a document as Document Graph (DG) [**19**] [**20**]. In the implementation described here, only two-kinds of relations, "*is a*" and "*related to*", are captured. Figure 2-3 shows an example of the concept graph built for the sentence "workers at an auto plant went on strike". In our implementation, the text documents will be converted into DGs by the gIG-Builders.

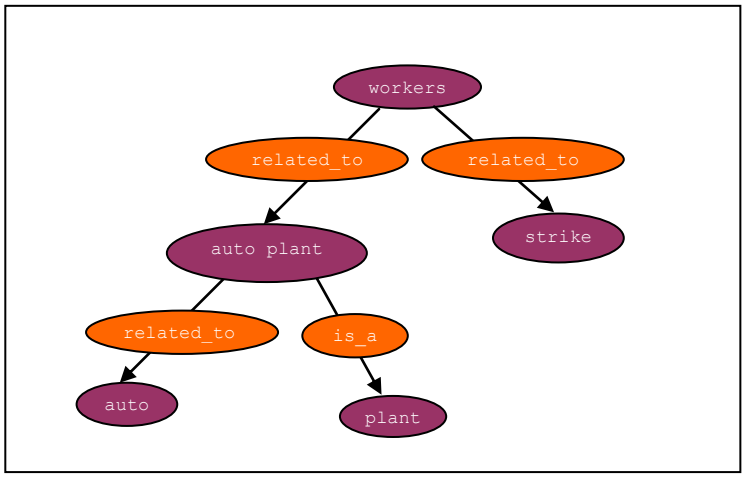

Figure 2-3 Concept graph

Resource Allocation

The partial relevancy values generated by the retrieval algorithm for a particular document are used to determine the amount of resource to be allocated to it. Priority (P ε [0,1]) of a document is the measure of its potential to be one of the relevant documents in the search space. Documents with higher priority values will be given precedence when allocating computing resources. In this particular implementation, we use the past partial similarity values of the document to calculate the priority. The priority (defined in [**24**]) of a document *d* is computed as follows:

$$P_k = \beta_1 q_1 + \beta_2 q_2$$

$$q_1 = \delta P_{k-1} + (1-\delta) Sim_k$$

$$q_2 = \begin{cases} \Delta_k & \Delta_k > 0 \\ -\Delta_k \dfrac{t_k}{t_0} & \Delta_k = 0 \end{cases}$$

$$\Delta_k = Sim_k - Sim_{k-1}$$

Equation 2-1 Priority metric for text retrieval

In Equation 2-1,

- P_k represents the priority of the document *d* at time *k*

- Sim_k is the similarity value of the document d at time k
- Δ_0 is the average expected similarity for the maximum parsing time
- t_k is the parsing time allocated for the document at time k
- δ, β_1, β_2 are the scaling factors

Additionally, we use a half-life decay function to scale the similarity value to incorporate the time critical nature of the information. The idea behind this is to give more preference to newly arrived documents. The half-life decay function has been used [**25**] to calculate the similarity of a document d at the kth step in the following way:

$$Sim_k = Sim_k . e^{-\frac{t}{h}}$$

Equation 2-2 Applying decay function in similarity metric

In Equation 2-2,

- Sim_k represents the similarity value of the document d at time k
- t is the total time for which document d has been processed
- h is the mean lifetime

2.5.4 Implementation Details

We now describe the implementation of text retrieval algorithm that was used within the I-FGM framework. The Java programming language was used to implement the various components and MySQL[13] database to store details of the documents such as document length, current value of the similarity, etc. The agents were implemented in a computing cluster with each agent being assigned a separate computing node. Each agent performs a function that remains unchanged throughout the simulation. The agents are autonomous and do not communicate with each other. The components of I-FGM are:

[13] http://www.mysql.com

1. **I-Foragers**: The first step in the processing cycle is to search for and download relevant documents from the internet. I-Foragers retrieve results from search engines such as Google[14], MSN (now known as Bing[15]), Yahoo[16], Looksmart and Teoma (now known as ask.com[17]). Each search engine is activated using a text query and top documents are downloaded. The search engines thus provide "fast and dirty" retrievals. Each search engine uses its own proprietary search method which has different performance levels for different types of queries. In order to quantify the performance of the search engine or any other third party retrieval method we use a metric called reliability.

   Reliability (defined in [**25**]) is calculated based on results from a number of test runs, using the following equation:

$$reliability = \alpha_1 \beta \frac{n}{k} + \alpha_2 s + \alpha_3 d$$

Equation 2-3 Reliability metric for text retreival

   In Equation 2-3,

   - A is the set of relevant documents for a query, $B \subseteq A$ is the set of relevant documents returned by the I-Forager
   - s is the average similarity measurement of documents in B
   - d is the average difference between rank of documents in B and the rank returned by the I-Forager
   - β is the ratio of the maximum number of documents gathered by any I-Forager to the number of documents gathered by this I-Forager
   - $n = |B|$
   - $k = |A|$
   - α_1, α_2, α_3 are scaling factors

[14] http://www.google.com
[15] http://www.bing.com
[16] http://www.yahoo.com
[17] http://www.ask.com

Using the reliability of the I-Forager, each document that is downloaded is given a measure called the first order expected similarity, which is calculated from the I-Forager measurements. For each downloaded document, it is a summation of the product of scaled reliability of the I-Forager and I-Forager measurement (which is the inverse of its rank in the search result that the I-Forager returned) over all I-Foragers. For a newly arriving document we set its initial priority according to two factors, one is its first order similarity and the other is the priority of its neighborhood. The neighborhood consists of the documents preceding and succeeding it in the current ranking of all the documents in the gIG-Soup (according to priority). The size of the neighborhood that should be considered while calculating the priority will depend on the reliability of the I-Foragers. For this we rely on a rule-of-thumb: higher the reliability, smaller should be the size of the neighborhood. The following formula [**25**] calculates the initial priority of the new document.

Initial priority P_{init} = max (first order similarity, average of neighborhood priority)

2. **gIG-Soup**: is used to store the documents, partially processed nuggets and also document attributes (document length, priority values, etc.). It should also be accessible to the other components of I-FGM. A shared folder using Network File System (NFS) provides the space to store the documents and nuggets. The advantage of using NFS is that many computing nodes hosting the different components of I-FGM can mount the shared folder and easily access the gIG-Soup. A relational database system MySQL[18] is used to store the document attributes such as length, priority, similarity, portion processed, location of the document in the physical disc, etc.
3. **gIG-Builder**: selects documents based on its priority value, processes it for a period of time and outputs the processed information as DGs. In our implementation we convert the text documents into DGs using techniques from Natural Language Processing (NLP) [**94**] to extract concepts (in the form of nouns) and relations from the text.

[18] http://www.mysql.com/

The components of the DG generator [**95**] are:

a. **Tokenizer:** Each sentence in the text is broken down into its constituent words and unimportant words, punctuations and symbols are filtered out.
b. **Parser:** A parse tree is produced from the text by using a Link grammar [**96**]. Although other kinds of parsers such as probabilistic parser [**97**] [**98**] are available, a link parser is used as it is efficient and easy to implement.
c. **Relations Generator:** The parse tree is used along with relations generators such as Noun Phrase heuristic (NP-heuristic), Prepositional Phrase heuristic (PP-heuristic) and Sentence heuristic (S-heuristic) to extract *is-a* and *related-to* relations from the text.
d. **Graph Generator:** The concepts and relations generated from the other components are then used to create the concept graph for the sentence. The concept graph is then merged with the DG already generated.

Ideally, the document with the highest priority value should be chosen for processing because they have the highest potential to be relevant. Due to the inherent uncertainty in prediction of future similarity value, it is difficult to define an optimum priority function. Hence we adopt the following document selection procedure with a random component. The top 10 documents are chosen from the gIG-Soup according to its priority values. A probability function that is biased towards the higher priority documents are used to select the document for processing. One of the main reasons for using this function is that many documents in the top 10 may have identical priority values and even when they are different they may only vary marginally. In such a case, following a strict selection of the top document will be inefficient. This was also shown in the experimental runs made on I-FGM. The document is processed for an amount of time called the parsing time which is calculated using its priority value.

4. **I-Matcher:** I-Matcher selects a document that has been processed by the gIG-Builder and compares the DG with the query graph and calculates the similarity value. The similarity between query graph q and document graph d_i is defined [**24**] below.

$$Sim(q, d_i) = \frac{n}{2N} + \frac{m}{2M}$$

Equation 2-4 Similarity measure for text retreival

In Equation 2-4, *n* and *m* are the number of concepts and relation nodes of the query graph found in the document graph. *N* and *M* are the total numbers of concept and relation nodes of the query graph. Note that two relation nodes are matched if and only if its parent and child concept nodes are also matched. In this equation, we treat relation nodes and concept nodes equally. The priority of the document is then updated using the similarity value.

5. **Blackboard**: The RDBMS database is repeatedly queried and the documents with the highest similarity values are displayed on the blackboard.
6. **Query Builder**: A natural language query is received from the user and the query graph is generated using the DG generator.

2.6 Summary

In this chapter, a detailed description of I-FGM was given along with a discussion of its successful implementation with text retrieval. The overhead incurred in the anytime text algorithm was kept low. The metric used for prioritizing the resource allocation was defined and a document selection strategy based on it was described. The principles of partial processing and the I-FGM architecture are leveraged in the next 2 chapters to design a real time image retrieval algorithm for large and dynamic search spaces.

Chapter 3

Partial Processing Based Image Retrieval

Although text retrieval with partial processing is an important first step, there is a need for further validation with other data types such as images. Also, text retrieval has been extensively studied and effective algorithms for interpreting text have already been developed. This is not the case for images where significant challenges in interpretation of visual information exist. Furthermore, partial processing will introduce an extra layer of complexity in the already difficult image retrieval process. On the other hand, recent advances in the field of semantic image retrieval, especially region-based image retrieval, has brought to the forefront algorithms that could potentially be used to interpret partial images. In this chapter, we discuss such a methodology for extracting and interpreting partial information from images. We validate the retrieval algorithms and demonstrate the efficiency of resource allocation strategies that were designed based on the partial interpretation of images. The results presented in this chapter have been published in [**26**] [**27**] [**28**].

3.1 Current Approaches

With the proliferation of images in web pages across the internet, searching for suitable images [**99**] [**100**] [**101**] has become an important research area in information retrieval. In this section, we will provide a summary of this research, with a focus on those techniques that will be later leveraged in this chapter. As such, the image retrieval research can be broadly classified into two subdivisions: Text Based Image Retrieval (TBIR) and Content Based Image Retrieval (CBIR).

3.1.1 Text Based Image Retrieval (TBIR)

Algorithms in TBIR [**102**] use the text annotation, HTML text or meta tags in order to search for relevant images. It is also referred to as Annotations Based Image Retrieval (ABIR). The earlier algorithms belong to this category and a survey on TBIR is given in [**103**]. Images are indexed based on the keywords in the

annotation and traditional text retrieval algorithm used to search for relevant images based on text based query. It is clear that the performance of the algorithms depends on the quality of the image annotation. Since annotation of images is a labor intensive task, it is not feasible for large and dynamic databases [**100**]. In web pages, the HTML texts [**104**] around the images provide additional description and context for the images and are widely used in commercial image search engines. Images such as satellites or medical images have text metadata. Metadata is a short introduction about a document, imagery, or database, describing its structure, content, and domain, and helps in deciphering its meaning. Methods use ontology [**105**] [**106**] containing specialized information, to extract semantic information from the metadata. Ontology [**107**] can be defined as a specification that consists of concepts and relations, and supports reasoning algorithms. TBIR methods for such specialized images suffer from the drawback that domain knowledge is required for effective retrieval and hence automation is difficult. In the face of such issues, there has been a gradual shift in focus from TBIR to CBIR algorithms that interpret the visual information in the image.

3.1.2 Content based Image Retrieval (CBIR)

CBIR is an important sub-area of image retrieval that aims to classify, identify and retrieve images based on its visual content rather than using human generated information in the form of meta-tags and annotations. Through the previous decade, there has been significant progress in this field that has been documented in various surveys [**100**] [**99**] [**108**] [**101**]. CBIR methods are categorized according to the type of search [**99**] such as image to image comparison, image categorization or relevancy ranking. In this work, we focus only on those aspects of CBIR that deal with relevancy ranking of images in the database for a given query. This is a classic information retrieval problem in which given a query, the top results are returned to the user. Although, a general solution for image retrieval still eludes researchers due to the difficulties posed by the well-documented phenomena of sensory gap and semantic gap [**29**] [**109**], CBIR techniques have shown great promise in automating the process of interpreting images. We decided to employ CBIR in I-FGM because other methods require extra-image information such as meta-tags or

HTML text from web pages containing the image. In the CBIR methods, the visual content or low level features of the images have to be converted into an intermediate mathematical form or signature before it can be interpreted. The choice of low level features and feature signature are important as it has a direct influence on the performance of the system. Examples of low level features [**110**] [**111**] in images are color, texture, shape and contrast.

Image algorithms may have global signatures or local signatures [**112**]. A global signature is a single quantity representing all the features in the image. Examples of global signatures include histograms [**113**] [**114**] and wavelets [**115**] [**116**] . With large databases, CBIR algorithms using global signatures produce a high number of false positives [**100**]. Hence, a more nuanced approach in feature extraction is required. Also, global signatures do not help in identification of distinct objects in the image and subsequently cannot help in extracting concepts and relations from the image. Extraction of concepts and relations are important as they help in understanding the semantic relationships between the concepts found in the images.

Region-based methods [**117**] [**118**] are a widely used type of CBIR that uses local signatures. Local signature represents a part of the image and multiple local signatures can be grouped together to represent a homogenous region. Region-based image retrieval have been used to reduce the semantic gap [**29**] that is more acute when global representation of the images are considered [**119**]. In these methods such as Blobworld [**120**], Integrated Region Matching (IRM) [**121**], NETRA [**122**], WALRUS [**123**] [**118**], images are divided into regions, each of which has similar pixel characteristics. This collection of regions forms a signature for the image. The premise behind these methods is that similar images will have similar regions. Methods such as WALRUS and SIMPLIcity [**124**] use wavelet transforms [**41**] to represent the feature signature of regions. They generate regions by moving a window over the image and calculating the feature signature for the part of the image under the window. These signatures are then clustered into regions using a distance measure. These methods work well even with complex images or scenes, containing multiple concepts or objects. One of the drawbacks in these methods is that they take

query images, instead of a text query. This prevents its direct usage within the I-FGM framework. Instead of moving or sliding windows, salient point extraction [**125**] can be used to identify important regions and calculate the feature signature of the regions using wavelet transforms.

We will use the image segmentation and region clustering techniques from WALRUS in our image retrieval algorithm. The feature vector that we use to represent the low level features of the image is similar to the one used in SIMPLIcity. Although SIMPLIcity is a well known regions method, it does only coarse grained classification of images into semantic classes such as "*outdoor*", "*indoor*", "*texture*", "*non-textured*", and "*photographs*". Our algorithm on the other hand deals with more fine-grained classifications. It identifies low level features by forming regions of similar pixel characteristics and then mapping them to the concepts. Although this is similar to the CAMEL [**126**] algorithm (a variant of the WALRUS method), there are significant differences. The concepts used in CAMEL are single object concepts such as "apple". It compares the feature vectors for this concept with incoming images and tries to determine if the concept is contained in the image using a distance based similarity measure. We found that CAMEL does not deliver the required performance when implemented with the images that we typically deal with. The main reason is that it is difficult to accurately identify single object concepts in images. Another important reason why we do not use this method is that it uses a distance based similarity measure that is adversely affected by the presence of non-relevant features in the images. A better option is to use machine learning techniques to match the low level features of concepts and images. ALIPR [**127**] [**128**] is a similar automatic annotation method that tries to map low level features to keywords. It is an established method that has given good performance and has also been used commercially. On the flip side, ALIPR cannot be used with the partial processing paradigm of I-FGM as the probability distributions associated with its feature vectors cannot be calculated with the incomplete image information.

Another technology that we will leverage in validating our algorithm is the web image search engine. We use the internet as our testbed since it is a rich source of images and dynamic in nature. We will use

image search engines such as Google and Yahoo[19] to do a first order filtering and narrow down the search space. These search engines cannot be used directly for real time retrieval as they use static indexes that are built offline [**129**]. Also, they require human generated information in the form of the HTML text surrounding the images [**104**] or meta-tags and cannot deal with images such as satellite imagery, aerial reconnaissance images, etc., which may not be accompanied by annotations.

3.2 Partial Processing based CBIR

The queries used with image retrieval algorithms may be text or image query. Retrieval based on image query converts the query and database images into a common mathematical signature and computes the similarity between the images usually with a distance based evaluation function. Using a text based query is more prevalent in image search applications and is also more challenging. It is challenging because it involves mapping the visual content of the images and the text content of the queries to a common representational form before comparison. Hence semantic interpretation of the visual content is required. Semantic gap [**29**] [**109**] is a major stumbling block in image interpretation/retrieval. With partial or incomplete image, interpretation of the images and similarity calculations becomes a challenge. This is because the interpretation can change widely as more information about the image becomes available. In the image algorithm that we have developed for partial processing, we use a combination of image sampling and machine learning techniques in order to mitigate the effect of partial information. We now describe the image retrieval algorithm that calculates the similarity using incomplete information that is incrementally gathered by partially processing the image. The work described in the following sections has been published in [**28**] [**27**] [**26**]

3.2.1 Detailed Description

Like a traditional CBIR algorithm, our methodology consists of extracting and identifying important features of the image. Our algorithm achieves incremental processing by considering only a part or window of the image at a time and progressively sliding the window over the whole image. The number

[19] http://www.yahoo.com

of windows processed at any one time is proportional to the processing time allocated. The pixel features of the image window are represented using the wavelet transform and forms the signature of the window. By clustering windows with similar characteristics, we form regions that correspond to concepts in the image. The centroids of the regions are combined to form the image signature vector. In order to use text queries, we have to convert the visual content of the image and the text content of the queries to concept graphs. As described before, concept graphs are the common knowledge representation used in I-FGM. Converting the visual information to text keywords or concepts is a challenge which is solved in our methodology by using a collection or library of image signature of commonly found concepts. A machine learning algorithm is used to match the signature of the image with those of the concepts in the library. The matched concepts are used to construct the corresponding concept graph representation of the image. The query is also represented as a concept graph and compared with those of the images using the matching algorithm from text retrieval to calculate the similarity value.

The image retrieval algorithm consists of following three processes:

a) Image Feature Extraction: Each image in the search space is processed to extract the visual information in the form of mathematical signatures. These signatures are then examined and interpreted. In our methodology, the pixel information of the images is mathematically represented by using wavelet [**41**] transformation. Since our aim is to extract concepts from images, we adopt the rule of thumb frequently adopted in region-based image retrieval algorithms that concepts corresponds to contiguous regions in the image and these concepts generally have similar pixel characteristics such as color and hue. We decompose each image into *l* regions by calculating the wavelet coefficients of a rectangular window of size m-pixels by n-pixels. The images are processed by moving this sliding window over the whole image, calculating the wavelet coefficients of each window and grouping windows of similar coefficients together to form regions. In order to reduce the variations in image interpretation in successive processing steps, we sample the windows from different sections of the image. This is a technique that will facilitate the quick formation of correct concept-regions mappings. Each image is represented by a 16 dimensional

feature vector (l=16) with each dimension representing a region of image. l=16 was selected as it has been shown to give good performance in other image retrieval algorithms [**130**] [**124**] . Each region is represented by a six dimensional vector representing the color and texture information of the images. LUV color space is used as it is a "perceptually uniform color space" [**131**] and most accurately models human perception of differences between colors and patterns [**131**]. LUV [**132**] also works well with distance metrics used in clustering algorithms and has been successfully used in other region-based algorithms [**119**] [**124**]. Unlike conventional color space such as RGB, LUV depicts all visible colors. There is also a straightforward conversion between pixel values in RGB and LUV. L denotes luminance while U and V encode the color information.

A mathematical representation for each window signature is calculated using the average values of the color channels in the image and the 2x2 Haar wavelet transform [**133**] on the L component. We chose Haar wavelets for computational efficiency [**118**] [**117**], performance and ease of implementation.. The signature of a window w is of the form [**28**] [**27**]:

$$h_w = \left\{ l_w, u_w, v_w, m^1{}_w, m^2{}_w, m^3{}_w \right\}$$

where l_w, u_w, v_w are the average values of the L,U,V color channels in a window w and $m^1{}_w$, $m^2{}_w$, $m^3{}_w$ are the high frequency components of the Haar wavelet transform of the L values in w.

A clustering algorithm is applied to the set of window signatures calculated from the image in order to identify regions with similar pixel characteristics. Clustering [**134**] [**135**] is an important and widely studied problem in data mining and unsupervised classification domains. Clustering algorithms aim to partition a data set of N points into K clusters such that the points within a cluster are more similar (or closer) to each other than points between two clusters. It is an unsupervised classification because neither the number of clusters nor the cluster membership of the data points is known before hand. The aim of a clustering algorithm is to minimize the distance between the points in a cluster and maximize the distance

between points in different clusters. This is a non-convex optimization problem [**136**] with the potential for many local minima that makes clustering a computationally intensive process. One way of classifying of clustering algorithms is based on the use or non-use of distance metric [**137**] [**138**] – probability-based or generative and distance-based or discriminative clustering. In probability based algorithms, instead of a fixed membership, each data point is given a probability measure of it belonging to a particular cluster. Typically, the probability distributions such as Gaussian distribution [**139**] [**140**] are fixed before hand for the data points and clustering algorithm determines the model parameters such as mean and standard deviation for each cluster. Probability-based clustering does not use a distance metric. Distance-based clustering use metrics such as Euclidean measure to define the distance between data points in a cluster. Clustering algorithms have also been categorized as partitioning and hierarchical clustering based on the way it forms the clusters. In partitioning algorithms, the data points are divided initially among the *K* clusters using some heuristic. The data points are then moved among the clusters to find the best assignment of the points to the clusters. In hierarchical algorithms, the final clusters are formed by a gradual process in which clusters are either split into smaller sub-clusters (divisional clustering) or sub-clusters merged (agglomeration clustering) to produce a larger cluster. In the agglomerative version of hierarchical algorithms, each data point has its very own cluster. The clusters closer to each other are then merged to form a larger cluster. This merging is continued until a threshold based on the value of K or the minimum intra-cluster distance is reached. In the divisional version, all points are assigned to a single cluster initially. Widely spaced data points in this mega cluster are assigned to separate clusters causing it to split into smaller clusters.

Among the many clustering algorithm available, we chose the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [**141**] algorithm primarily for its ability to work with incremental data without a large overhead. BIRCH is an agglomeration based hierarchical clustering algorithm that is designed to work with large datasets and takes memory and I/O usage into consideration while clustering. One of the drawbacks of BIRCH is that it can work only with numeric data. It is able to work with large,

incremental data by creating a summary of the data points that reflects the distribution of values in the data space. Clustering is performed based on the summary rather than the actual data points, allowing it to deal with large datasets efficiently. The summary is stored in the specialized data structure called the CF-Tree (Clustering Feature Tree) which also models the hierarchical relationship between clusters and their constituent sub-clusters. The tree has two factors: 1) Branching Factor (B), and 2) Threshold (T), which determine the height of the tree. Each non-leaf node has a set of Clustering Features (CF) for each sub-cluster that is represented by the node. CF is a 3-tuple (N_i, LS_i, SS_i) where N_i is the number of data points in the cluster, $LS_i = \sum_{i=1}^{N_i} X_i$ and $SS_i = \sum_{i=1}^{N_i} {X_i}^2$. Each leaf node contains the actual cluster points. It may be noted that when a new data point or a new sub-cluster have to be added to the tree, the CF metrics are used to decide on the points of insertion instead of the actual cluster points. This is how BIRCH algorithm is able to deal efficiently with incremental clustering.

The area (covered by the data points) and the centroids of each cluster regions are calculated from the output generated by BIRCH. The algorithm takes the number of clusters as one of its parameters and this is set at 16. The centroids are arranged in a decreasing order with respect to the area of their respective clusters and are combined to form the final image signature as depicted in the following equation [**27**] [**28**]:

$$f_x = \left\{ l_{x,1}, u_{x,1}, v_{x,1}, m^1{}_{x,1}, m^2{}_{x,1}, m^3{}_{x,1}, \ldots\ldots\ldots\ldots, l_{x,16}, u_{x,16}, v_{x,16}, m^1{}_{x,16}, m^2{}_{x,16}, m^3{}_{x,16} \right\}$$

where $l_{x,i}$, $u_{x,i}$ $v_{x,i}$ are the average color values in the centroid for region i and $m^1{}_{x,i}$, $m^2{}_{x,i}$ $m^3{}_{x,i}$ are the wavelet transforms for the L color space in the centroid for region i.

In summary, images in the search space are processed incrementally. An image is selected for processing and computational resources are allotted to it. Based on the processing time, image extraction algorithm is applied to a particular number of windows. These features are added to the existing clusters of the image

and the new clusters with centroids are calculated. The centroids of the clusters are then used to create the new image signature.

b) Creation of the Concept Library:

As noted before, the concept library contains representative image signatures of concepts that are used to identify concepts in images from the search space. In order to produce such a representative signature, we must choose an appropriate image depicting the concept and use its wavelet signature. Due to the ambiguity in accurately defining visual concepts, calculating such representative signatures is a daunting task. For example, for a concept "car", what would be an appropriate image – should it be a blue car or a red car? Should it be a Mercedes or a Honda? As we can see, many images can represent a single concept. For most real world concepts, it is difficult to find one perfect image to represent the concept. Hence we should use a learning algorithm that uses multiple training images or instances containing the concept to calculate a classifier for the same. This algorithm should also be able to work with the ambiguities created by the difference in the features for images representing the same concept. Learning techniques can be broadly classified as Supervised, Unsupervised and Reinforcement Learning. In supervised learning such as decision tree [**142**], the training instances used are labeled correctly without any ambiguity and this is used to calculate the classifier that will use this model on the test data. In unsupervised learning techniques such as clustering [**135**], the training instances do not have any labels and the learning algorithm tries to find the underlying structure of the training data and uses it to classify any new instance. Although reinforcement learning [**143**] also uses unlabeled training data, it also gets additional information in the form of delayed rewards in order to perform the classification. Since the ambiguities in our training data is not as extreme as unsupervised or reinforcement learning methods, we use a modification of supervised learning called the Multi-Instance Learning (MIL) framework [**144**] [**145**]. In MIL, instead of labeling each training instance, a set of instances or a *bag* is given a common label. It uses positive and negative bag of instances in order to create its classifier. A bag that contains at least one

instance that is true to the given label is called a positive bag otherwise it is a negative bag. In image retrieval, the application of MIL has been studied with respect to scene classification [**146**].

Specifically, we use logistic regression within the MIL framework to train the classifier. Logistic regression algorithm has been used in MIL learning [**147**]. Another reason for using a statistical learning technique to identify similar features in images is that methods using Euclidean distance have been found to be lacking in sufficiently modeling human perception [**100**] of visual information. Logistic Regression (LR) [**148**] [**149**] is appropriate for matching concepts in the concept library with images because the decision variable for each concept has two values: 1- concept is found in the image 0- concept is not found in the image. LR has been found to work well with such dichotomous decision variables [**150**]. The performance of LR has been compared with other popular classifiers used in machine learning such as Tree Induction [**151**], Naïve Bayesian [**152**] and Support Vector Machine [**153**]. In comparison with tree induction, LR performs better with small training data set while Tree Induction is better with a large data set [**151**]. The drawback of Naïve Bayesian classifiers is that it requires conditional independence among variables, thus limiting its applications. LR is found to perform more accurately than Naïve Bayesian [**152**]. With respect to Support Vector Machine, the loss function used in LR has been shown to be approximated by a Support Vector Machine [**153**] [**154**] and therefore they have a comparable performance. Learning based on LR has found applications in CBIR [**155**] and in relevance feedback [**156**] for image retrieval.

The concept library plays a critical role in mapping the low level features of the images (image signature) to high-level text based concepts. In order to achieve this, the library stores the signatures of all the concepts that are likely to be found in the images. Each concept signature is in the form of logistic regression classifier generated by processing the image signatures of the training images. The image signature for each training image is extracted using the image extraction procedure. The training image set of a concept consists of positive and negative images. The positive images depict the concept whereas

the negative images do not. The image signatures are used to create a logistic regression classifier for the concept. Each training input is of the form [**28**] [**27**]:

$$t_j = \{l_{j,1}, u_{j,1}, v_{j,1}, m^1_{j,1}, m^2_{j,1}, m^3_{j,1}, \ldots, l_{j,16}, u_{j,16}, v_{j,16}, m^1_{j,16}, m^2_{j,16}, m^3_{j,16}, r_j\}$$

where $l_{j,i}$, $u_{j,i}$ $v_{j,i}$ are the average color values in the centroid for region *i* in the training image *j*. $m^1_{x,i}$, $m^2_{x,i}$ $m^3_{x,i}$ are the wavelet transforms for the L color space in the centroid for region *i* in the training image *j*

$$r_j = \begin{cases} 0 & for\ negative\ images \\ 1 & for\ positive\ images \end{cases}$$

The logit function for an image vector *x* is given as [**28**] [**27**]:

$$g(x) = \alpha + \beta_1 l_{x,1} + \beta_2 u_{x,1} + \beta_3 v_{x,1} + \beta_4 m^1_{x,1} + \beta_5 m^2_{x,1} + \beta_6 m^3_{x,1} + \cdots + \beta_{91} l_{x,16} + \beta_{92} u_{x,16} + \beta_{93} v_{x,16} + \beta_{94} m^1_{x,16} + \beta_{95} m^2_{x,16} + \beta_{96} m^3_{x,16}$$

where α, β_i ($1 \leq i \leq 96$) are weights calculated by the maximum-likelihood method from the training images. The logistic regression algorithm is implemented using the WEKA software [**157**].

The library is created in an offline process and is depicted in Figure 3-2. The concepts in the library are accompanied by a short, manually created textual description. The text description is used to generate the concept graphs using the text parser and graph generator described in Chapter 2. The concept graphs of all the concepts identified in the image will be combined to create its final concept graph representation.

c) Image Matching: The image signatures generated by partial processing are compared with the concept classifiers in the library for matches. This process is shown in Figure 3-1. The matching algorithm use logistic regression to determine the likelihood that a particular concept is contained in the image. The concept annotations for all such relevant concepts are combined together to form the image description. The description is converted in to a document graph and compared with the query graph and the similarity is calculated using Equation 2-4. The similarity values are then used to determine the set of relevant images and for allocating resources.

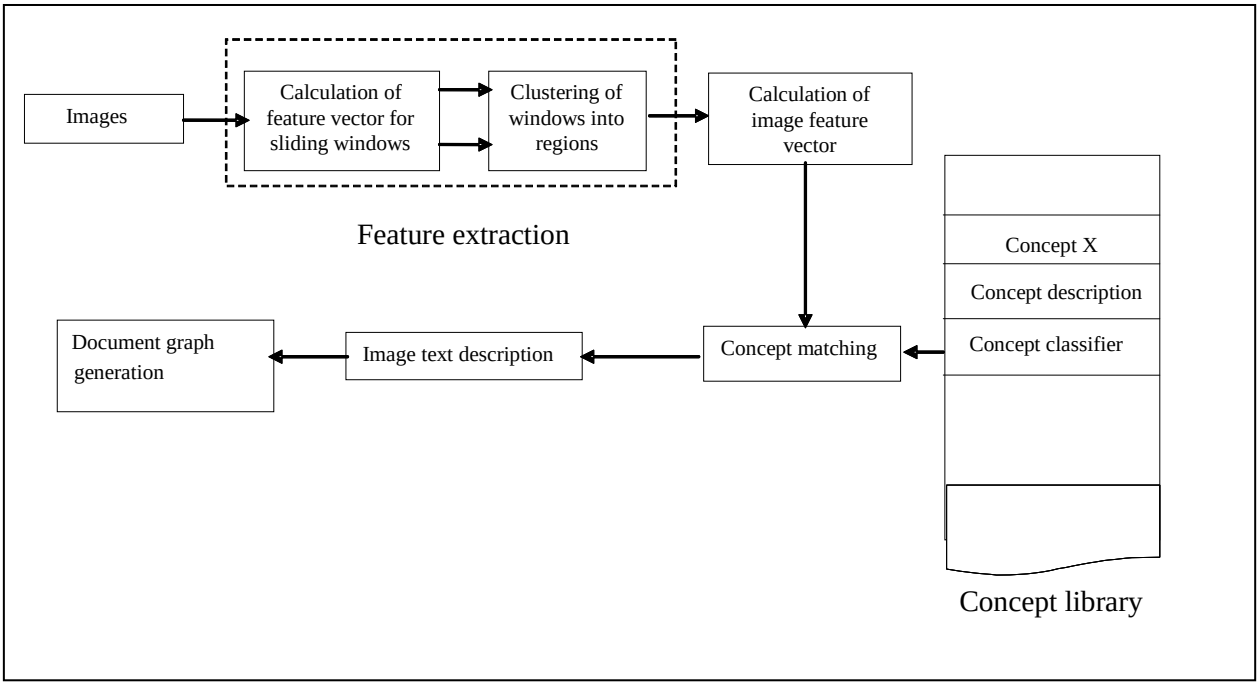

Figure 3-1: Extraction of relevant concept from the image

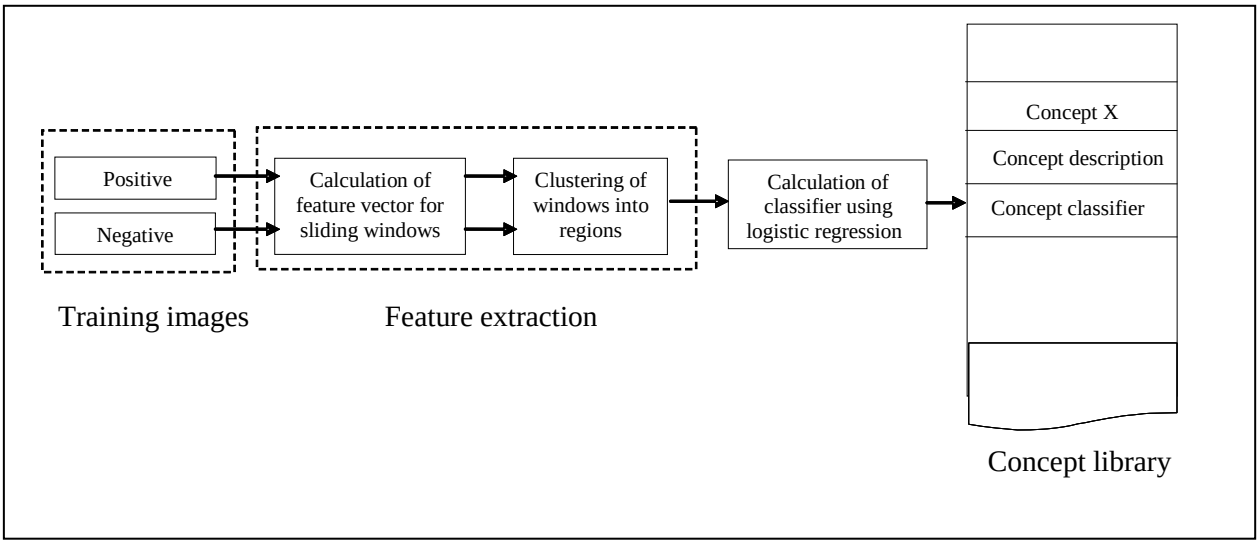

Figure 3-2: Offline creation of concept library

3.3 Summary

In this chapter, we designed an image retrieval algorithm that uses parts of the image to determine its partial relevancy values. The images are incrementally processed and a clustering algorithm is used to

create regions. The new partial information in the form of feature signatures, generated from successive processing is used to update regions or create new ones. We also showed how the low level features can be used to identify higher level concept and how this is used to generate the concept graphs. The concept graph representation enables us to use natural language queries. In the next chapter, we will validate the image retrieval algorithm with experimental results.

Chapter 4

Validation of Real Time Image Retrieval Algorithm

In the previous chapter, we proposed an image retrieval algorithm that incrementally processes images in a large and dynamic search space and uses the partial-values to guide resource allocation and produce quicker results. Experimental validation of the image algorithm is required to show its viability. The validation experiments should not only test the retrieval power of our algorithm but also its effectiveness in real time retrieval. The work described in the following sections has been published in [**28**] [**27**] [**26**]

4.1 Experimental Validation

Since some features of our algorithm such as wavelet coefficients have already been validated in other region-based CBIR methods, our validation procedure focuses on the new techniques such as concept matching and resource allocation strategy. We deploy the image algorithm within the I-FGM framework and test it using web images related to disaster relief scenarios as the testbed. Our validation process can be divided into two stages 1) precision analysis, and 2) real time recall analysis. Precision and recall [**158**] are widely used performance metrics in information retrieval. The precision analysis is used to determine how good the image retrieval algorithm is with respect to retrieving relevant images. Since there are many methods in CBIR and it is not possible to compare with each of the methods, we compare against a popular wavelet method called WALRUS [**118**] [**123**]. WALRUS, which is representative of the region-based image methods, is also a proven CBIR algorithm. The feature extraction process and window signature used in our algorithm is similar to WALRUS. Therefore, by comparing our performance against WALRUS, we show the utility of mapping low level features to higher level concepts. In the real time recall analysis, we validate the hypothesis that using resource allocation strategy based on partial

relevancy values will lead to faster results. For this, we use two control systems that do not use the partial similarity values to do resource allocation.

4.2 Implementation of Partial Processing based CBIR in I-FGM

The implementation procedure is similar to text retrieval. Network File System (NFS) directories and MySQL tables are used for storing documents and their header information in the gIG-Soup. The agents are coded in JAVA for portability, flexibility and ease of implementation. The points of difference from the text retrieval include: 1) using image search engines within I-Foragers, and 2) implementing the image algorithm within the gIG-Builders. Since the similarity measure is measured by comparing the query graph and the document graph generated from images, the I-Matcher is similar to that of text retrieval.

a) I-Forager: Web image search engines are used to retrieve the initial set of images from the internet. These search engines perform the initial filtering of the documents. The three I-Foragers utilize Google, MSN and Yahoo search engines, which accept natural language query, to download the top 50 results into the gIG-Soup. Using the ranking provided by the search engines for each document, the I-Foragers calculate the first order similarity values.

b) gIG-Soup: The gIG-Soup functions as the central data repository of I-FGM for holding the newly retrieved and the partially processed images. In order to support distributed processing, the gIG-Soup is implemented using a NFS directory. The NFS directory helps to efficiently share the documents in the gIG-Soup among the agents that are implemented on multiple machines. gIG-Soup also consists of a MySQL database that stores the document details such as image file name, similarity values and first order similarity. The agents use the details in the MySQL to select a document for processing. As the documents are processed, the values in the MySQL are updated. Since the database helps in sharing the information about the documents processed by multiple agents, it helps in asynchronous agent communication.

c) gIG-Builder: The image retrieval algorithm described in Chapter 3 is implemented in the gIG-Builder. Agents that function as gIG-Builders incrementally process image documents and convert the visual information into a document graph. It selects an image document based on the priority values. The current status information of the document such as current similarity, number of windows processed, etc. is accessed from the database. Based on the processing time allocated for the image, gIG-Builder will process a certain number of windows. 8x8 pixel windows and a window overlap of 4 pixels are used. For each window, the wavelet based signatures are calculated and clustering is performed. If the image has been already partially processed, the new window signatures are added to existing clusters and re-clustering is performed. We employ the BIRCH clustering algorithm with a threshold value of 70. The newly updated cluster centroids are matched with the classifiers in the concept library to identify image concepts. Using the description of the concepts identified in the image, a document graph is generated.

d) **I-Matcher**: The document graph generated by the gIG-Builders is compared with the query graph in the I-Matcher. The similarity value for the document is calculated by comparing the document and query graphs using the formula [**27**] [**28**] in Equation 4-1.

$$Sim(q, d_i) = \left(\frac{n}{2 * N} + \frac{m}{2 * M}\right) * \rho$$

$$\rho = \begin{cases} 1 - \dfrac{C_1 - C_2}{L} \; where \; C_1 \geq C_2 \\ 1 - \dfrac{C_1}{L} \; where \; C_1 < C_2 \end{cases}$$

Equation 4-1 Similarity metric for image documents

In Equation 4-1, q is the query graph, d_i is the document graph of the image document. n and m are the number of concepts and relation nodes respectively, that are common to the query graph and document graph. N and M are the total numbers of concept and relation nodes of the query graph. Two relation nodes are matched if and only if their parent and child nodes are matched. The similarity measure used here is identical to the one in text retrieval except for the term ρ. ρ is the probability measure of the

confidence in the similarity and is related to the number of image concepts identified. It also helps to scale the similarity measure to deal with the uncertainty brought on by the semantic gap. C_1 is the number of concepts matched to the image and C_2 is the number of possible relevant concepts. We calculate C_2 from the number of concepts in the query graph. For some images, the matched concepts vary widely. By using a confidence measure, we are quantifying how sure the machine learning algorithms is about its matches. If the matches are small in number, it means that the image has certain unique pixel characteristics that are found in only certain concepts. The confidence measure will be high for these images. For images that match a large number of concepts, we say that the matching algorithm is not so confident about its matches.

After determining the new similarity value after each partial step, the I-Matcher re-calculates the priority value for the image. The priority measure is the crucial metric that guides the allocation of computing resources to the documents. The order in which the documents are selected for processing and the amount of processing are decided using the priority metric. The priority formula is identical to the one used in text retrieval and the weights specific for images are used in the formula. These weights are determined by running some initial tests on image testbeds.

e) **Blackboard**: In order to display the changing values of the top documents in the search space, the Blackboard is used. At regular intervals, the Blackboard queries the database in the gIG-Soup in order to determine the top documents and this is relayed to the user. Since the partial processing methodology does a refinement of the partial similarity values of the documents, the current results displayed by the Blackboard gradually converges to the actual list of relevant documents.

4.3 Experimental Validation

We validate the image retrieval algorithm in two steps. The first step in our validation procedure is to test the precision performance of the image algorithm. Precision [**158**] is a common IR metric which is defined as the ratio of the number of relevant documents retrieved to the total number of relevant

documents. We compare the precision results of the image algorithm with those of WALRUS algorithm. We will henceforth refer to our image algorithm as Concept-based algorithm to distinguish it from the WALRUS-based algorithm. In the second step, we validate that resource allocation using the partial similarity values generated by the image algorithm will lead to efficient and quick retrieval of images. We compare its performance with two control systems called Partially-Intelligent (PI) and Baseline systems that use other resource allocation strategies.

4.3.1 Control Systems

In order to evaluate the performance of the image retrieval algorithm with partial processing, we compare it to a set of control systems that have the same architecture as I-FGM but differ in the way the systems selects the documents for processing. The components in all the systems have the same functionality as in I-FGM. The two control systems are:

1. **Baseline**: In this control system, the gIG-Builders pick a document randomly and process it for a fixed amount of time. I-Matcher calculates and updates the similarity as the processing proceeds. Priority is not used in Baseline system. This control system is used to test if the multiagent architecture is sufficient to provide superior performance and if intelligent allocation of computing resources is really required.
2. **Partially-Intelligent**: The system uses the initial priority of the documents to rank the documents in the gIG-Soup and selecting it for processing. This priority is not refined as the processing progresses. This control system is used to check if priority refinement is necessary or if using only the measures returned by the I-Foragers are enough.

Though both Baseline and Partially-Intelligent systems perform partial processing, they do not utilize the partial similarity values generated. We employ partial processing in Baseline and Partially-Intelligent systems in order to provide real time display of results. If the Baseline and Partial-Intelligent systems had to wait for all the documents to be processed, it would never perform better than the I-FGM system.

4.3.2 Testbed

In order to conduct the validation experiments, we build image testbeds using web images downloaded from the internet. We select the scenario "Natural Disasters" as the domain from which queries are selected. The concept library is also populated with concepts related to this scenario. For each query, we construct a testbed of images by downloading the top 50 results from image search engines such as Google, Yahoo and MSN. The set of 5 queries selected for this scenario are:

1. Building damaged by hurricane Katrina.
2. Firefighters fight wildfires.
3. Heavy snow storms in the winter.
4. Houses damaged by tornado.
5. Houses damaged by tsunami.

4.3.3 Creation of Concept Library

Basing all the queries on a particular scenario helps in simplifying the construction of the concept library. Using general queries would require the concept library to contain concept from a vast number of domains. Instead, we tailor our library to contain concepts that are related to the "Natural Disasters" scenario.

Each concept in the library is accompanied by a short description and a classifier. Examples of concepts and description are given in Table 4-2. The classifier is calculated using multi-instance training with positive and negative images. The training images are also web images downloaded from the internet. The positive and negative training sets consist of 20 images each. The images are processed to extract the wavelet based features and clustered to generate the centroids. The centroids of the positive and negative images are then used to create the logistic regression classifier. The feature extraction and the training process are explained in detail in Section 3.2.1. The concepts used for the experiments are listed in Table 4-1.

Beach	Damaged Buildings	Cold weather	Dinosaur	Firefighter
Fires	Fruits	Tribal People	Snow Storm	Mud Slide
Mountains	Trees	Floods	Forest Fires	Rainstorm
Tornado	Elephants	Flowers	Food	Tsunami
Hurricane	Katrina			

Table 4-1 List of concepts in the concept library

	Concept	Concept description
1.	Beach	Beach has sand. Beach has lines with coconut trees. Beach has a blue ocean. People sunbath on the beach. Waves can be seen on the beach. People lie on the beach. People watch boats on the beach. There are boats on the beach. It is windy on the beach. People relax on the beach. People play on the beach.
2.	Damaged Buildings	Damaged buildings are caused by natural disasters. Damaged buildings are caused by flood, storm, and hurricane. Damaged buildings have damaged walls, leaking roots, broken doors, broken windows. Damaged buildings can also be caused by terrorism attack. For example, World Trade Center was damaged by terrorist attack.

Table 4-2 Examples of concepts and concept description in the concept library

4.3.4 Performance Metrics

The performance metrics used in evaluating the systems are:

1. **Recall ($Recall_d(t)$):** is defined as the ratio between the number of relevant documents retrieved by time *t* and the total number of relevant documents. It is a commonly used metric in IR [**158**]. We calculate the recall values from the system snapshots and compare their plots. We say the system has achieved total recall when its recall value is 1.0, i.e. it has successfully retrieved all the relevant documents.
2. **Document-Wait Time (W_d):** is calculated for each relevant document and is defined as the amount of time in seconds that has elapsed before the document *d* appears on the Blackboard for the first time. We calculate the document-wait times for all relevant documents in each system and compare them in a bar chart.

4.3.5 Validation of Image Retrieval Algorithm

In order to validate the image retrieval algorithm, we will do a precision analysis of the Concept-based image algorithm. We will compare its performance with a baseline in the form of the WALRUS algorithm. A WALRUS-based partial processing image algorithm has been implemented and tested within the I-FGM framework in [**26**]. Although the Concept-based and WALRUS-based image algorithms have common features (e.g. wavelet feature signature) formal testing is required to demonstrate the viability of using our image algorithm. We also seek to show that our algorithm performs at least as well as WALRUS. As such the points of difference between WALRUS-based and Concept-based algorithms are: 1) Concept-based algorithm uses text based query while WALRUS requires image queries, and 2) WALRUS directly compares the feature signature of the query image with the search space images while Concept-based algorithm maps the image signatures to higher-level text-based concepts using machine learning techniques. This allows the Concept-based algorithm to work with text queries.

$$Similarity(P,Q) = \frac{area(\cup_{i=1}^{n}(P_i)) + area(\cup_{i=1}^{n}(Q_i))}{area(P) + area(Q)}$$

Equation 4-2 Similarity metric for WALRUS-based Algorithm

As in the Concept-based algorithm, the WALRUS-based algorithm creates regions by clustering the window signatures. It compares the regions of the query image with those of the search space images using a distance metric, and generates a list of region pairs that are similar or close to each other. The metric used to measure the similarity in the WALRUS-based algorithm is given in Equation 4-2 [**26**] where P and Q represent the retrieved image and the query image respectively. The set of ordered pairs $\{(P_1, Q_1), \ldots, (P_n, Q_n)\}$ form a similarity region pair set for P and Q, where P_i is similar to Q_i and for $i \neq j$, $P_i \neq P_j$, $Q_i \neq Q_j$.

The WALRUS-based and Concept-based I-FGM prototypes are run with the testbeds of 5 queries. Although the classic definition of precision [**158**] of a retrieval algorithm is the ratio of the relevant documents retrieved to the total documents retrieved, we use a slightly modified definition. Precision is defined as the ratio of the relevant documents among the top n retrieved documents. The two systems are run on each of the query testbed and the top 15 images for each query are analyzed manually to determine their relevance and finally calculate the precision.

	Precision	
	WALRUS-based Algorithm	Concept-based Algorithm
Query 1	9/15	10/15
Query 2	10/15	9/15
Query 3	9/15	11/15
Query 4	13/15	12/15
Query 5	10/15	10/15

Table 4-3 Precision values attained by WALRUS-based and Concept-based I-FGM systems

Results Analysis

The top 15 images retrieved by each system are analyzed manually to determine whether they are relevant or not. After determining the relevant images, the precision values are calculated for each query. By analyzing the results given in Table 4-3, we see that both systems have similar performance. Concept-based system does better for queries 1 & 3 and the WALRUS-based system does better for queries 2 & 4. So on the whole, each system wins over the other in 2 queries and have a tie for query 5. Thus we have been able to validate that the Concept-based algorithm is indeed viable as it was able to perform at least as good as an established algorithm like WALRUS. Similar performance results for both WALRUS and Concept-based IFGM show that the performance is not degraded when concept matching are used with the feature extraction. The advantages of our algorithm, in its ability to work with text query and identify

high level concepts even with partial information, are crucial for a unified ranking for search spaces with heterogeneous documents.

4.3.6 Evaluation of Resource Allocation

The aim behind developing our image algorithm is to perform real time retrieval. The partial processing allows for the design of intelligent resource allocation strategies for quick and efficient image retrieval. We validate this by implementing three controls system that use the image algorithm with different types of resource allocation strategies and comparing their performance. The system that does intelligent resource allocation using the priority metric is labeled as the I-FGM system. We compare it with two other control systems: 1) Baseline, and 2) Partially-Intelligent (PI) systems.

Before the control systems are run on the image testbeds of the 5 queries, the relevant set of documents have to be selected. All the documents are processed using the image algorithm and similarity values are calculated. The top *n* documents with the highest similarity values are selected for each query as its relevant set. The size of the relevant set is kept small (around 10). The size varies from query to query so as to make sure that no non-relevant document has the same similarity value as a relevant document. The metrics that are used to compare system performance are: 1) recall, and 2) document-wait time. Recall values at regular intervals over the simulation period is noted and a graph of Recall vs Time is plotted. The recall values show how quickly a system discovers all the relevant documents and the document-wait time indicates how quickly the system was able to zero into individual documents in the relevant sets. The aim of the resource allocation strategies in the control systems is to minimize the time to achieve full recall and minimize the document-wait times of the relevant documents.

Result Analysis

The consolidated performance results of the control system are given in Table 4-4, which list the number of documents that each system discovers for each query. I-FGM does better than the other control system for 2 queries (queries 4 & 5) and a tie with Baseline for query 3. Baseline and Partially-Intelligent

performs the best for one query each – query 1 and query 2 respectively. Detailed analysis is done by looking at the recall vs time plots (Figure 4-1, Figure 4-3, Figure 4-5, Figure 4-7 & Figure 4-9) and document-wait time bar charts (Figure 4-2, Figure 4-4, Figure 4-6, Figure 4-8 & Figure 4-10). In the recall vs time plots, the recall values are along the y-axis and time in seconds along the x-axis. In the document-wait time graphs, the documents are labeled on the x-axis in the form *x.y* where *x* is the search engine that retrieved the document and *y* is the rank given by the search engine.

In the Recall vs Time plots, the superior performance of I-FGM is not as obvious as in those of text retrieval validation experiments. I-FGM recall has the best performance in query 4 (Figure 4-7) where it reaches most recall points earlier than other systems. In query 5 (Figure 4-9), although it discovers most documents earlier than other systems, its recall values are better than other systems only in the range [0.2, 0.6]. Query 2 (Figure 4-3) is a query where one system, namely Partially-Intelligent clearly dominates. In other queries, clear dominance by a system is not seen in the recall graph. One reason for this fluctuating performance of the systems is that the similarity measures of the images vary widely in terms of its rate of change, making it difficult to predict future values. In text retrieval, the partial similarity values generated by the concept graph method vary less rapidly. This is easier to predict, leading to better resource allocation. Designing a better priority function that uses a stochastic model of the similarity function is required and is part of future work.

By analyzing the individual documents in the document-wait time graphs, we see that the superior performance of PI system for query 2 (Figure 4-4) is due to the fact that the relevant documents are relatively highly ranked by the search engines. Since PI system uses the static first-order similarity measure to prioritize resource allocation, it will do well in such circumstances. It may be noted that despite query 2 being an ideal case for PI system, I-FGM system does almost as good as PI by retrieving 3 documents as against PI's 4 documents. I-FGM and Baseline systems have a tie for query 3 (Figure 4-5, Figure 4-6) by discovering 4 documents each while PI system discovers 1 document. On analyzing the three documents (Yahoo.41, Google.53, Google.35) discovered by Baseline system, we see that its initial

partial similarity values are close to the final similarity values. This means that these documents are identified as relevant with minimal processing. Also, they are lowly ranked by the I-Foragers and have a low initial priority, leading to a long wait before the I-FGM system selects them for processing. I-FGM clearly does better in query 4 & 5. For query 4 (Figure 4-7 & Figure 4-8) and query 5 (Figure 4-9 & Figure 4-10), I-FGM is the clear winner. For query 4 it discovers 5 documents against Baseline's 3 and for query 5, I-FGM again gets 5 while both Baseline and PI get 4 each.

	No. of Relevant Documents	**Baseline**	**Partially-Intelligent**	**I-FGM**
Query 1	11	7	3	2
Query 2	10	3	4	3
Query 3	**9**	4	1	4
Query 4	**8**	3	0	5
Query 5	11	4	4	5

Table 4-4 No. of documents retreived first by each system

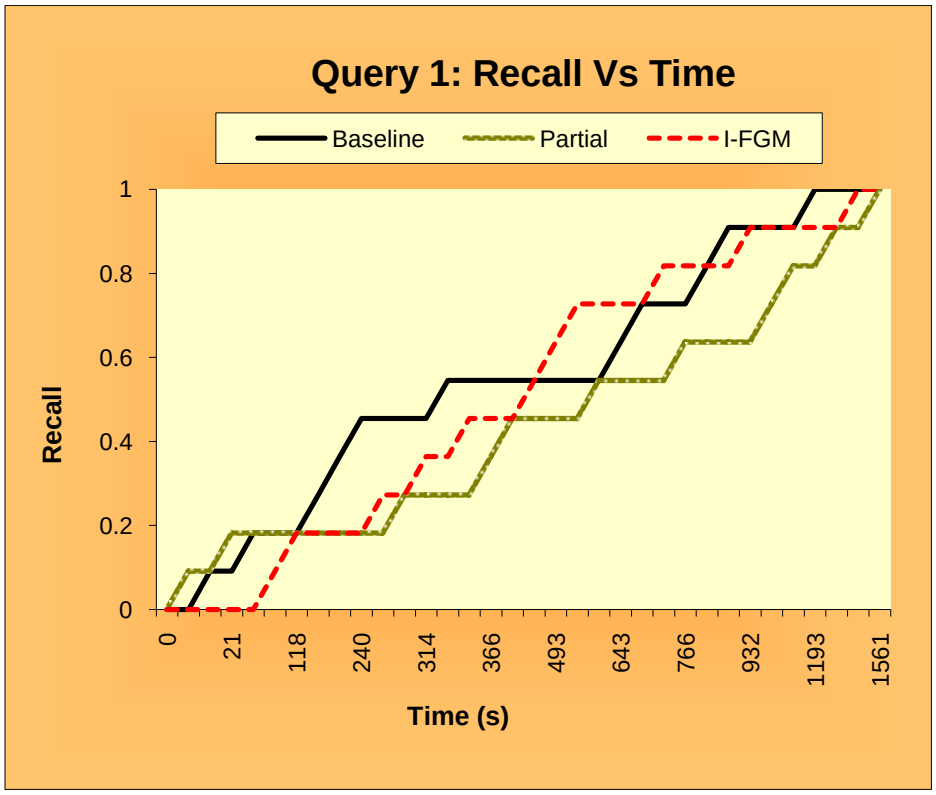

Figure 4-1 Recall vs Time for query 1

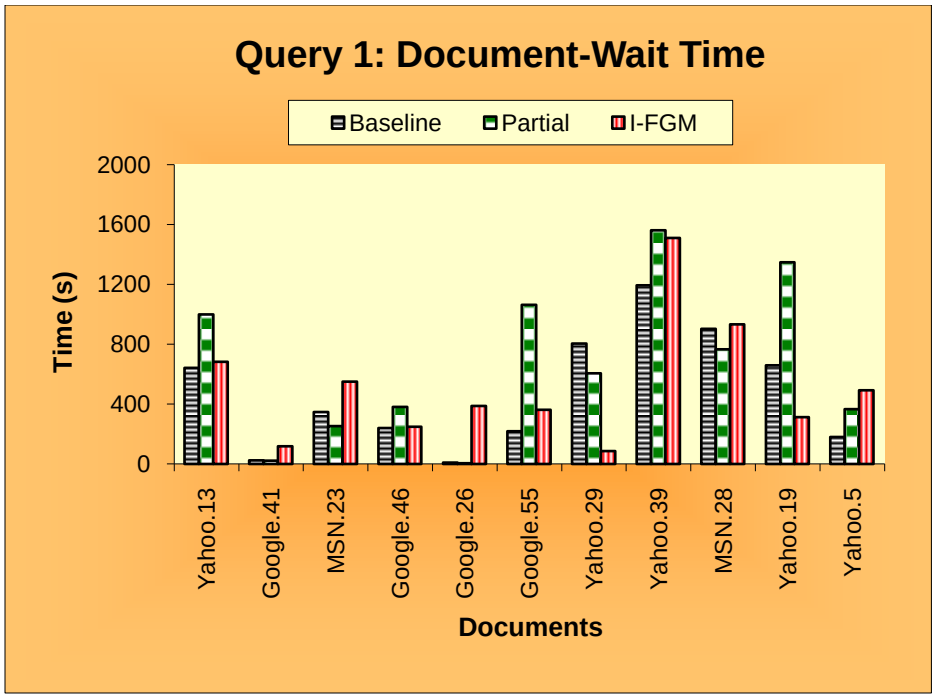

Figure 4-2 Document-wait time for query 1

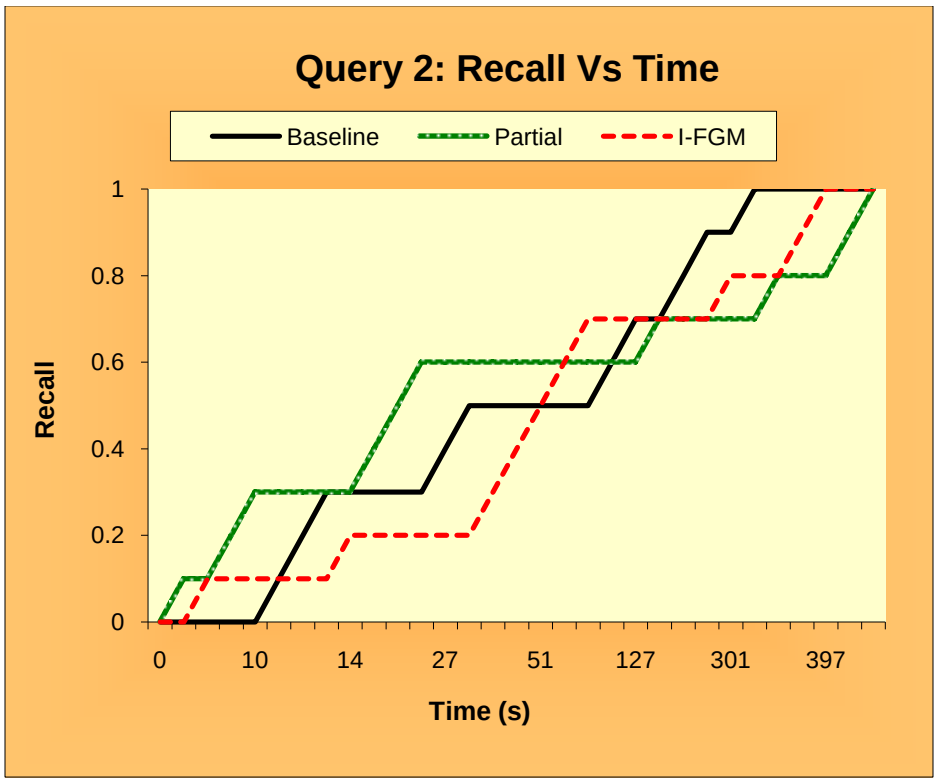

Figure 4-3 Recall vs Time for query 2

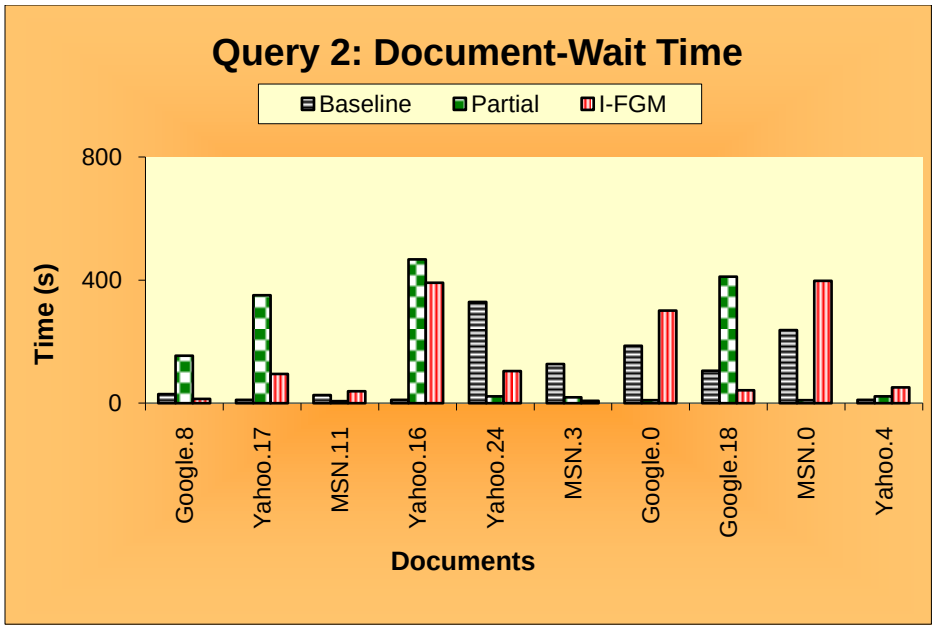

Figure 4-4 Document-wait time for query 2

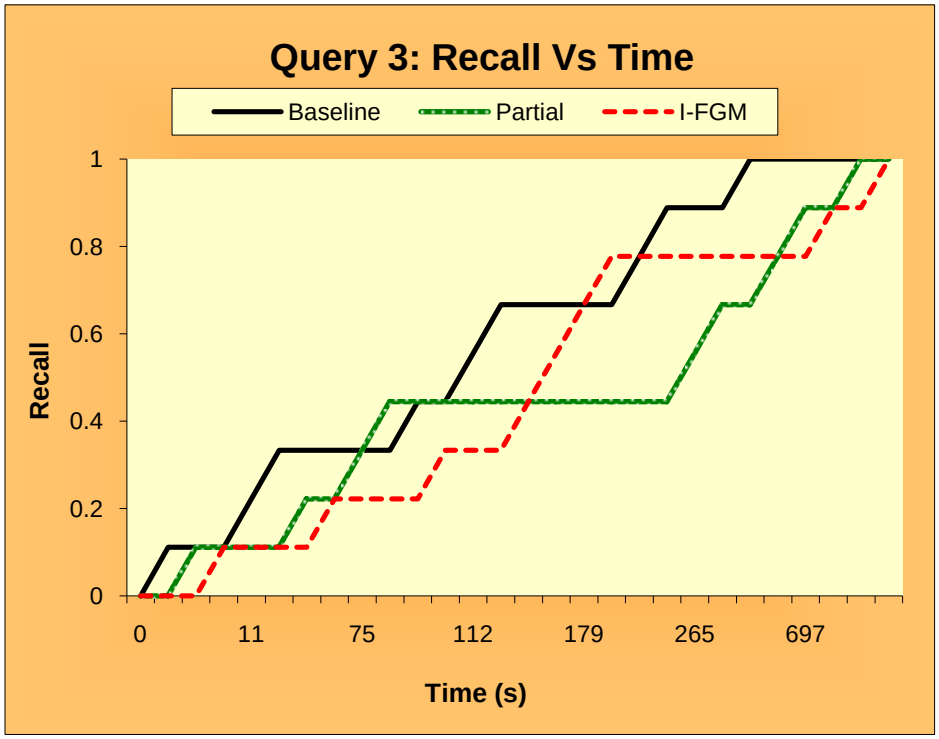

Figure 4-5 Recall vs Time for query 3

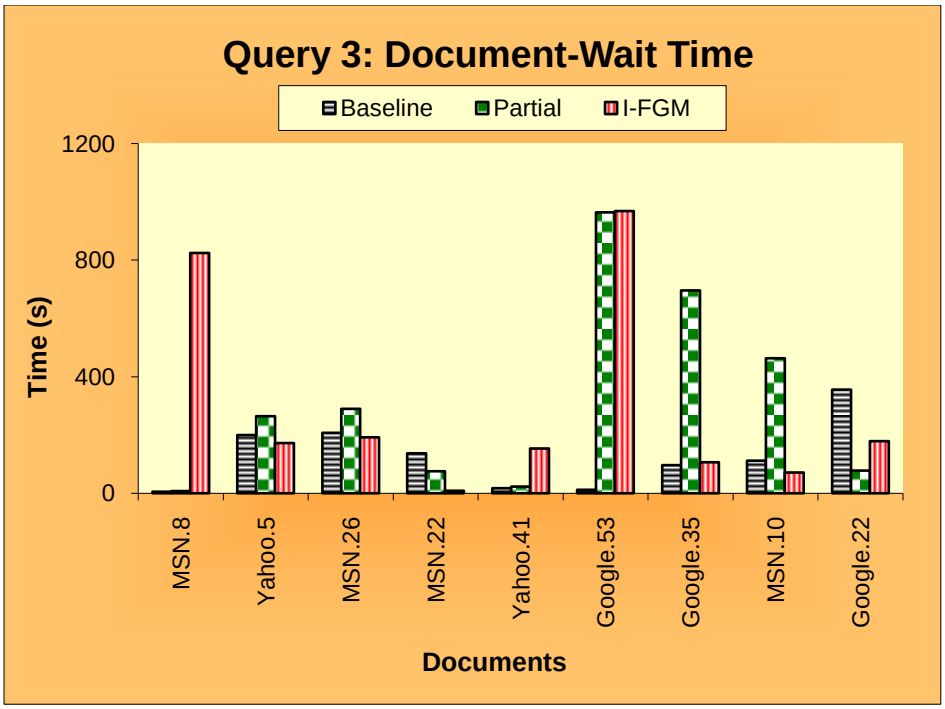

Figure 4-6 Document-wait time for query 3

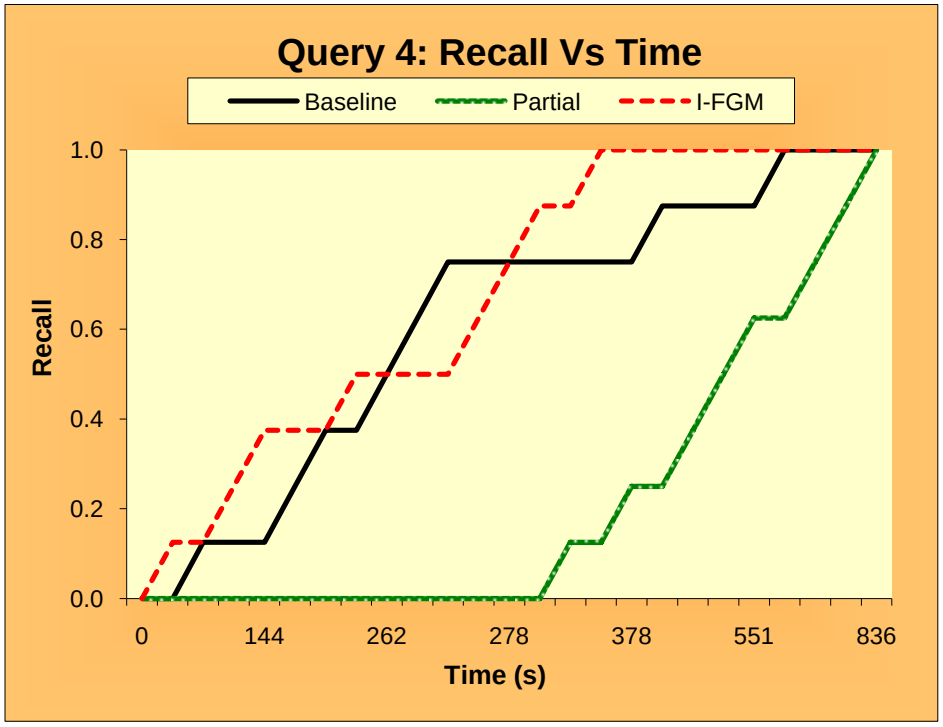

Figure 4-7 Recall vs Time for query 4

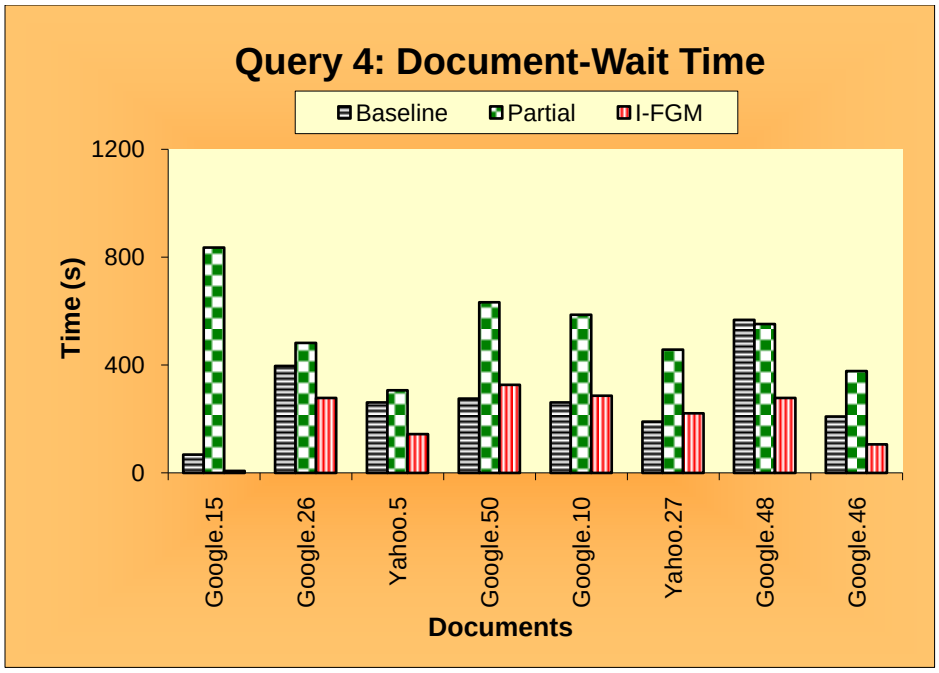

Figure 4-8 Document-wait time for query 4

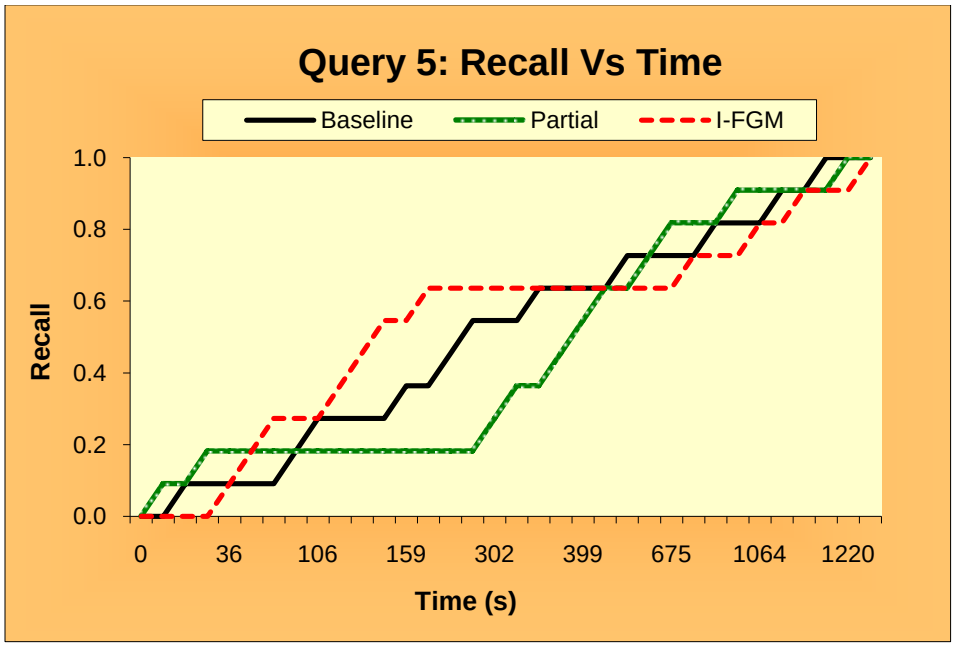

Figure 4-9 Recall vs Time for query 5

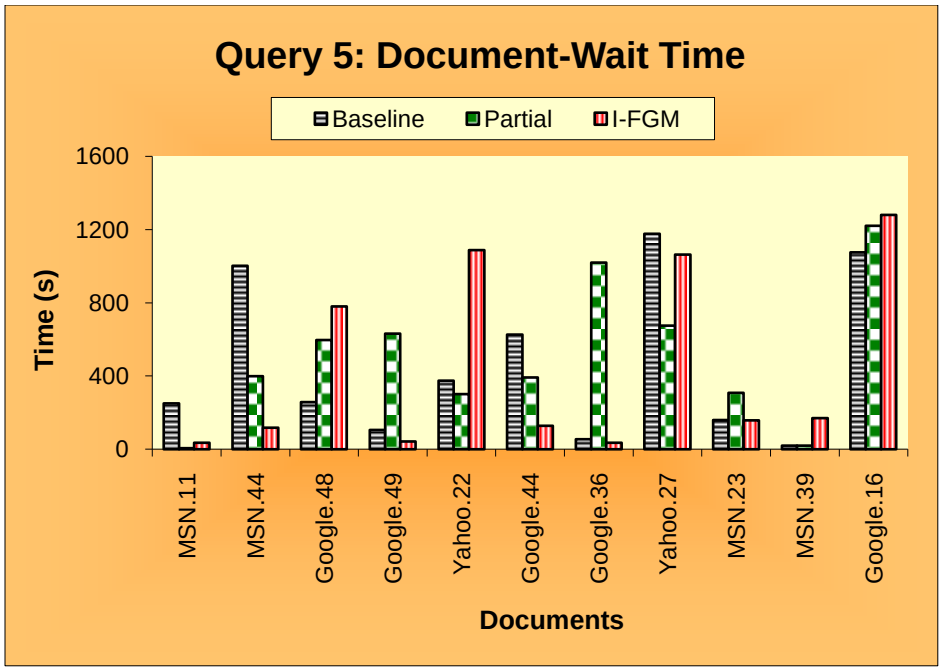

Figure 4-10 Document-wait time for query 5

4.4 Summary

We performed experimental validation of the image retrieval algorithm that was designed to work with partial processing. We formulated a resource allocation scheme to work with partial similarity values and tested it against control systems representing conventional processing paradigms. We showed that using partial-values helped in getting the results faster than the conventional methods. We have been able to demonstrate that even for a complicated data type such as images, the partial processing paradigm can be successfully applied for real time retrieval.

Chapter 5

Resource Allocation Framework for Partial Processing

5.1 Introduction

The main focus in the chapters 3-4 has been to demonstrate the viability of using anytime retrieval algorithms within the partial processing paradigm for real time retrieval. We did this by showing that similarity values can be calculated even from the partial information generated at each processing step and that these partial-values can be refined efficiently to get the final values. Although we have successfully demonstrated viability, another aspect of partial processing that is crucial for generating results in a time bounded manner is resource allocation. Designing efficient resource allocation strategies for agents to choose a particular document or candidate and to allocate a particular amount of computing time to it in a large search space is an interesting research problem. It is important to have a formal framework that will model the decision-making process and interaction of the agents processing the documents. In contrast to other search problems in large and dynamic search spaces, the monotonically non-decreasing sequence of the partial-values generated by the anytime algorithms gives the search space a unique structure and the dynamism a certain flow that should be leveraged in the model. In this chapter, we propose a multiagent based resource allocation framework for multiagent systems that models these unique characteristics and acts as a foundation on which sophisticated resource allocation algorithms can be designed. We begin by discussing some of the issues and challenges that the framework should address followed by a survey of current multiagent decision making methodologies. We then provide a discussion on the details of our framework. The work presented in this chapter has been published in [**45**].

5.2 Issues and Challenges

The greater issues of size and dynamism of the search spaces create challenges such as resource constraints, partial-observability and non-determinism that have to be taken into account by resource allocation methodologies.

1. **Complex Environment:** With respect to resource allocation, the environment of an agent consists of the partial-values of the documents in the search space. Based on these values, the agents have to make a decision on selecting a particular document or candidate. Due to the large number of candidates, the number of partial-values that the agent has to consider is also large. Additionally, the search space of partial-values changes as the candidates are processed or as new candidates are added. The size and dynamism of this search space of partial-values makes it a "complex environment" [**159**] for the agents to operate in. A complex environment has the following characteristics:
   a. Partial-Observability [**21**]: Due to its limited computational resources, the agents can only observe or capture a subset of the environment at a time. Since it is not possible to model the entire search space, this limited view of the search space is used for making decisions on what candidates to process, which agents to communicate with, etc. To add to these challenges, even the limited view of the search space may be corrupted by noise.
   b. Dynamic: The environment is constantly changing due to the addition of new candidates, removal of fully processed candidates or the actions of other agents. This leads to change in the parameters of the state space model such as state variables, goal states etc.
   c. Non-Deterministic [**143**]: An environment is said to be non-deterministic when the outcome of a particular action is not guaranteed. Since the search space is dynamic, the effect of actions does not remain constant but may vary with time. Due to partial-observable nature of the environment, it may not be possible to accurately gauge the effect of actions. This makes it difficult to assign preference values to agent actions.

2. **Resource Constraints:** In a distributed processing environment, each agent has access to only a limited amount of processing power, communication bandwidth and memory. Scaling multiagent solution to large, real time search problems is indeed a challenge. The decision making and interaction process in the agents have to be designed taking the resource constraints into account.
3. **Partial-Values:** The partial-values calculated by the partial processing provide a structure or ordering to the search space that can be exploited by the resource allocation. This is due to the monotonically non-decreasing quality of the partial-values. However, using the partial-values is not straight forward as they are only approximations of the final values. The ordering of the candidates changes as their values are refined; creating an extra level of dynamism and this has to be taken into account by resource allocation. We label this as internal dynamism to contrast it with the traditionally defined dynamism or external dynamism. External dynamism is caused by the entry and exit of candidates and agents in the search space. It is clear that internal dynamism changes the environment that an agent works in and has to be modeled to make effective decisions on resource allocations.

5.3 Background

Since the goal of this chapter is to formulate a resource allocation framework that will support existing resource allocation algorithms, it is important to survey the current methodologies. We will limit our discussion to those methods that deal with large and dynamic search spaces. In addition to describing them, we will point out their strengths and weaknesses.

Decision making for multiagents has been studied in the artificial intelligence domain under the topic "Multiagent Planning" [**42**]. In a multiagent system, an agent works in coordination or in competition with other agents. The decision making of the agents is modeled in term of states and actions. Based on the information from other agents and sensory input from the environment, the agents must choose actions to achieve a set of goals. Planning algorithms may be broadly classified as reactive and deliberative planning. Reactive planners [**160**] use only the current information about the environment to make decisions while a deliberative planning agent also takes into account future states of the

environment. Reactive plans consist of a set of rules that map states to actions. Reactive planners select an action based on the current conditions. A "state" is the snapshot of the environment at a time instant and is quantified in terms of parameters deemed important to the problem being solved. In classical deliberative planning research, the problem [**161**] is formulated as a 3-tuple (I,G,A) where I is the set of initial states, G is the set of goal states and A is the set of actions. The deliberative planning algorithms aim at determining the sequence of actions to change the initial state to a goal state and this sequence of actions is termed as the agent plan or policy.

Classical planning algorithms make certain assumptions about the environment that limits its applicability in the real world. The environment is assumed to be fully observable and the state the agent is in, can be accurately sensed by the agent. The actions are considered to be deterministic – an agent in state s performing action a will always transition to state s'. In the real world, planning algorithms have to deal with issues of partial observability, non-determinism and dynamic environments.

Markov Decision Process (MDP) [**162**] [**143**] is a decision-theoretic framework that deals with non-deterministic environments by modeling contingencies. The possible outcomes of executing action a at state s are called contingencies and the MDP models use probability distributions for all possible actions for each state. Based on the principle of maximizing utility, actions are taken at each state that leads to a transition between states which is represented by the transition probability. Formally MDP is defined by a 6-tuple (S,A,T,R,h,γ) . S is the set of states of the environment, A is the set of actions taken by an agent or controller, T is the transition function between states, R is the set of rewards, h is the planning horizon and γ is the discount factor. The transition function T is formulated in terms of conditional probability as $T(s,a,s') = T(s'|s,a)$ which measures the probability of transition to state s' when action a is taken at state s. Rewards R encode the preferences of the agent and $R(s)$ represents the reward gained by an agent in state s. Due to Markov property, the action taken by the agent depends only on the information modeled in the current state and not on previous states. An important parameter in the planning process is the planning horizon h which is the amount of future steps that need to be planned for by the agent when

selecting an action a in state s. For infinite-horizon planning problems, a discount factor $\gamma \epsilon [0,1]$ provides a bound on the possible future rewards. A policy $\pi: S \rightarrow A$ is a mapping of states to action and $\pi(s)$ represents the action that the agent will take at a state s. The objective of methodologies using MDP is to formulate an optimal policy π^*. The average value of the reward that is gathered by the agent in state s by following a policy π, is given by $V_\pi(s)$ and is calculated as follows [**42**]:

$$V_\pi(s) = R(s) + \gamma \sum_{t \epsilon S} T(s, a, t) * V_\pi(t)$$

Due to the recursive nature of the value function V_π, policy iteration procedures for finding the optimal policy in MDP problems are ideal candidates for employing Dynamic Programming [**163**] techniques.

Although MDP models deals with non-determinism in the environment, partial-observability of the environment is a critical issue that still needs to be overcome especially when the state space is large. The agent may only be able to observe a part of the search space and this truncated view needs to be explicitly modeled. Partially Observable Markov Decision Process (POMDP) [**21**] [**164**] [**165**] is a variation of MDP that models the uncertainty in observing the states of the environments. The agent is assumed to have sensory input of only a limited number of parameters and as such uses a limited representation of the state of environment. The snapshot of the sensory inputs is called an observation and each observation z corresponds to an actual state s. POMDP models are represented using 8-tuple (S,A,O,T, Z, R, h, γ) [**166**] which is similar to the representation for MDP except for the additional sets O and Z . O and Z represent observation function and observation states respectively. For each observation $z \epsilon Z$, action $a \epsilon A$ and state $s \epsilon S$, the observation function $O(z, a, s)$ defines the probability that the agent will make an observation z when it executes action a to transition to state s. Additionally, belief state $b(s)$ is used to represent the probability that the POMDP system is actually in state s. Typically, the transition probabilities and rewards for a MDP problem are initially unknown. Reinforcement Learning (RL) [**143**] [**167**] [**168**] is an AI technique that uses the interactions of the agents with the environment to learn the transition probabilities and construct reward structure of the problem.

Now that we have introduced the popular tools such as MDP, RL and POMDP that are used in multiagent decision making process, it may be noted that these methods were originally designed to work with static environments. The policies are generated offline and then applied to the decision making process without the possibility of making any future refinements. Although these methods have strong theoretical underpinnings, they can be applied efficiently only to a domain with small and stationary state space. In a stationary state space the probability distribution of various parameters such as the transition function, observation function and state variables do not change over time. Even with simplifications such as limited horizon planning, it has been shown that finding an optimal policy in POMDPs is PSPACE-complete [**169**]. Variations of these methods that use approximate policies and compact state representations have been suggested for real time search in large and dynamic environments. One such method is called factored MDP [**170**] [**171**] in which the environment is modeled with the help of a set of random variables $X_1, X_2 \ldots\ldots X_n$ and each state is a joint instantiation of these variables. Using the principle of conditional independence, transition and observation functions can be compactly represented within probabilistic structures like Dynamic Bayesian Network (DBN). Instead of looking at all possible state transitions, factored MDPs allow for subsets of important states and actions to be considered allowing for faster policy generation. Another compact state representation technique is state aggregation [**172**] [**173**] wherein states that have similar reward functions and actions are grouped together and policies are generated for these aggregated states.

MDP and RL based algorithms have also been developed for real time applications [**174**] where the policies have to be generated within strict deadlines. Online planning algorithms have been proposed for POMDPs that use approximate offline policy to guide the online policy generation. Approximate offline algorithms [**174**] [**175**] include blind policy methods [**176**] [**177**], point iteration methods [**178**] and approximate MDP methods [**179**]. In blind policy methods, the agents generate a set of policies by "blindly" executing the same action at every state. These policies are generated quickly and provide lower bounds for future policies for that environment. In approximate MDP methods, we ignore the partial-

observability in the model design and formulate the decision making process as a traditional MDP process. Other online planning methods work by interleaving short planning phases with execution [**43**]. These methods have a limited planning horizon to cope with the dynamism in the search space. Such planning methods fall under the category called continual planning [**180**] [**181**]. During execution phase, these online planning algorithms monitor the effectiveness of the plan and may undertake plan modifications if necessary.

A hybrid planning architecture that interleaves deliberative and reactive planning [**182**] has been proposed as a way to deal with a dynamic environment. Contingent on the conditions in the environment, the planning process switches from being more deliberative when the environment is changing slowly to being more reactive when it is changing fast. One of the ways of implementing time bounded deliberative planning is by using the framework of anytime algorithms. In these anytime planning methods [**183**], the plan generation algorithms provide valid plans. The quality of the plans improves as the time for planning is increased.

One of the drawbacks of the planning algorithms described above (especially deliberative planning) is that they assume that all the states, actions and events can be anticipated in the beginning and explicitly represented. In many real world problems, the state space is initially unknown and the agent has to explore the environment along with the decision making process. Such agents are also called situated agents [**184**]. Situated agents do not perform long term planning and have a simple decision making process that use local information and interactions with the environment in order to select the immediate action. Real time heuristic algorithms [**185**] for situated agents in real time path planning have been developed. These algorithms are modification of the classic A* search algorithm [**64**]. The unknown terrain of the robots is divided into grids and the search algorithm helps in the time bounded calculation of the utility of moving into the neighboring grids. Hence it achieves real time search capability by limiting its search horizon.

5.3.1 Summary

We surveyed some of the relevant methodologies for multiagent decision making in complex and dynamic environments. Rapidly changing state space and goals prevent the usage of traditional MDP methods that employ offline generated plans or policies. Online planners interleave deliberative and reactive phases to deal with the changing search space. Also, traditional methods assume that the search space is known in the beginning which is not true for the information spaces that we are considering. We also see that agents that employ reactive planning and evolve the decision rules by interacting with the environment are able to fare better in real world problems.

5.4 Multiagent based Resource Allocation Framework

The framework (Figure 5-1) that we will present now is aimed at modeling the important characteristics of the search space of partial-values that are generated by partial processing. We have designed the framework to be component based with each component focusing on one aspect of the resource allocation process. The components and their interactions are designed to support plug-n-play, allowing for changes to be made to one component without fundamentally changing the others. The cornerstones of the framework are the search space model and explicit representation of different types of dynamism in the search space.

5.4.1 Framework Design

In order for the agent to make decisions, it is important for them to have knowledge of the search space. A search space model is required that will compactly represent the constituents and variations in the search space. It is the crucial lynchpin that ties the other components of the framework together. The search space model should enable quick selection of the best candidate for processing. The two main challenges that prevent this are: 1) search space is large, and 2) partial-values are only approximations whose reliability changes. Intelligent selection requires sorting the large search space and selecting the candidates with high partial-values. Since the order of the candidate will change with time, the sorting has

to be performed. This is neither efficient nor scalable. Our search space model solves this problem by grouping the candidates into bins with each bin containing candidates with similar partial-values. The model uses the fact that each candidate has two variables: 1) portion of the candidate that has been processed $r_d \epsilon [0,1]$, and 2) partial-value $v_d \epsilon [0,1]$. These variables change as the candidate is processed. A grid based division of the search space, depicted in Figure 5-2, is used. Each grid-block in the grid space holds candidates within a certain range of r_d , v_d values. As these values change, the candidate is relocated to a new grid-block. Due to the monotonically non-decreasing values of v_d and monotonically increasing values of r_d, the movement of a candidate within the grid is not erratic but follows a general upward trajectory. By grouping the candidates in grid-blocks, the search space can be compactly represented using grid-block specific parameters such as number of candidates in a grid –block, number of agents in the grid-block, etc. The agents also move between grid-blocks. The candidates and agents get their positional coordinates in the grid by virtue of their membership in a grid-block. By using distance metric such as Euclidean distance or Manhattan distance with these positional coordinates, it is also possible to create neighborhoods of different sizes for a grid-block. The neighborhoods make the decision making process scalable in two ways: 1) depending on their computational capacity and time constraints, agents can limit their modeling of the search space to immediate neighborhoods, and 2) agents in a grid-block can reduce communication costs by interacting only within its neighborhood. By representing the flow of candidates and agents between the grid-blocks, we are able to model the dynamic changes in the search space. Hence compact space representation, variable neighborhoods and dynamism modeling are the main advantages of the search space model.

The framework is depicted in Figure 5-1. As such, the different components of the framework are:

1. Search Space Model
2. Interaction Module
3. Prediction Module
4. Internal Model

The internal model encompasses the decision making process that determines the behavior of the agent. The internal model [**186**] may consist of: 1) beliefs that the agent have of the state of the environment and other agents, 2) goals that the agent is trying to achieve, and 3) possible actions that it can take. Agents communicate with each other to share information about their environment and goals, which help in coordinating their actions. Agent communications are facilitated by the interaction module and the information gathered is used to update the search space model. In addition to these components, the framework has a prediction module that contains algorithms for approximating future values of the search space variables. The algorithms in internal model and interaction module also use the variables represented in the search space model. We now describe each of these components and their sub-components in detail.

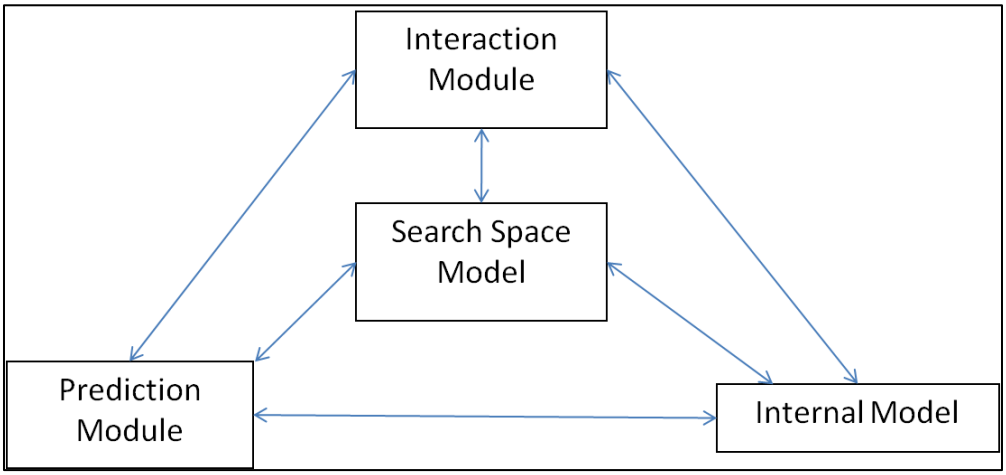

Figure 5-1 Agent model in the resource allocation framework

5.4.2 Search Space Model

The search space is modeled as a Cartesian space of 2 dimensions with each candidate d having two variables: 1) portion of the candidate processed or process-ratio $r_d \in [0,1]$, and 2) current partial-value of the candidate $v_d \in [0,1]$. We use a grid-based approach for modeling this search space by dividing it into m x n grid-blocks with m grid-block along the x-axis and n grid-block along the y-axis (Figure 5-2). Here the search space is said to be of dimension (m,n). Each grid-block is denoted by $B(i,j)$ where i, j are its

coordinates in the grid and $1 \geq i \geq m$ and $1 \geq j \geq n$. Based on its values of r_d and v_d, a candidate d is assigned to a specific grid-block. A candidate d in grid-block $B(i,j)$ has process-ratio r_d and partial-value v_d where $\left((i-1)*\frac{1}{m}\right) \leq r_d < \left(i*\frac{1}{m}\right)$ and $\left((j-1)*\frac{1}{n}\right) \leq v_d < \left(j*\frac{1}{n}\right)$. Additionally, we define two sets of grid-blocks $T_B$and F_B called top-grid-blocks and full-grid-blocks respectively. $T_B = \{B(x,y)\}$ where $1 \leq x \leq m, y = n$ and $F_B = \{B(k,l)\}$ where $k = m, 1 \leq l \leq n$. It may be noted that grid-blocks in T_B have candidates with the highest partial-values and grid-blocks in F_B have candidates with the highest process-ratio. An agent a can be assigned to only one grid-block at any one time and this grid-block is termed as its home-block $H_a(i,j)$. The agents can only process the candidates in their home-block. Reallocation of computational resources occurs when agents leave their home-block, move to a new grid-block and select candidates in the new home-block. Using grid-blocks, the candidates are grouped into smaller, more homogeneous groups. We consider the candidates in a grid-block to be homogeneous, as they have similar values of r_d and v_d and have high probabilities of ultimately achieving similar final values. Based on the values of r_d and v_d, it is reasonable to assume that candidates with low r_d and high v_d values have a higher chance of ending up in the relevant set of candidates. Using this assumption, the grid-space model can be divided into regions of "exploration" and "exploitation" based on the quality of candidates and their potential to yield high final values. Another benefit of using grid-block is that dynamism in the search space can be modeled in terms of grid-block changes due to variations in the number of candidates and agents, and their resulting flow between grid-blocks.

Exploration and Exploitation

The notion of exploration and exploitation are prevalent in the field of combinatorial optimization especially for metaheuristics [**187**] such as genetic algorithms [**188**]. In the context of genetic algorithms, exploration is concerned with increasing the diversity of the population by adding new candidates while exploitation deals with reducing the diversity by keeping only the candidates with high fitness values and removing the rest. In our grid-based model, it can be intuitively seen that the candidates in certain grid-blocks have a better chance of being relevant than candidates in other grids. Hence agents processing

candidates with high partial-values and low process-ratios is said to do exploitation as these candidates have a higher probability of being relevant. Similarly, agents working on candidates with low partial-values and high process-ratios are said to perform exploration. In order to quantify this explorative/exploitative nature of the candidates in a grid-block $B(x, y)$, we use a metric called the risk parameter $R_{risk}(x, y)$. It is based on the probability of the candidates in $B(x, y)$ to reach a grid-block in the top-grid-blocks T_B, weighted by the remaining unprocessed portion of the candidates. For each grid-block $B(x, y)$, we define two sets of grid-blocks: $T_B(x, y)$ and $F_B(x, y)$. $T_B(x, y) = \{B(i, j)\ \forall i, j : x \leq i \leq m,\ j = n\}$ is a sub-set of T_B that can be reached by the candidates in $B(x, y)$. $F_B(x, y) = \{B(i, j)\ \forall i, j : i = m, y \leq j \leq n\}$ is a sub-set of F_B that can be reached by the candidates in $B(x, y)$. In a grid space of dimensions *(m, n)*, $R_{risk}(x, y)$ is formally defined as:

$$R_{risk}(x, y) = 1 - \frac{|T_B(x, y)|}{|F_B(x, y)| + |T_B(x, y)|} * \left(1 - \left((x - 1) * \frac{1}{m}\right)\right)$$

Equation 5-1 Risk parameter

The normalized risk values of a 4x4 grid-space using Equation 5-1 are given in Table 5-1. All grid-blocks $B(x, y)$ with risk parameter $R_{risk}(x, y) < 0.7$ are designated as exploitative and are marked in red. The explorative regions are represented in blue and have $R_{risk}(x, y) \geq 0.7$ respectively. Agents switch between processing explorative and exploitative regions based on the dynamism conditions in the grid and the quality of results generated. It may be noted there that the metric $R_{risk}(x, y)$ does not have a fixed definition in the search space model and our formulation is only one of the ways to model the explorative and exploitative behavior.

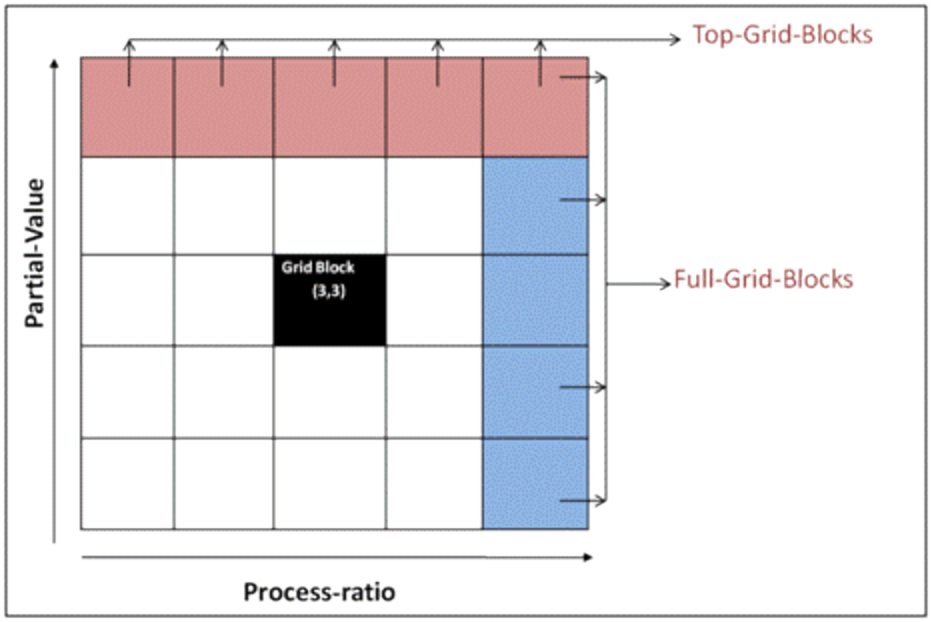

Figure 5-2 Grid search space model

0.21	0.46	0.71	0.93
0.35	0.58	0.79	0.97
0.45	0.66	0.84	0.99
0.53	0.72	0.87	1

Table 5-1 Risk parameter values for 4x4 grid

Internal and External Dynamism

One of the novelties of the search space model is the explicit categorization of dynamism as internal and external dynamism. External dynamism can be defined as the change in the search space caused by the entry or exit of candidates or agents from the search space. A candidate is said to exit the search space when it has been fully processed and its final value is known. As the partial-values of the candidates change, they are moved to new home-blocks to conform to the grid search space model. Similarly, the agents may also move between different grid-blocks. We model this movement of agents and candidates using graph-theoretic techniques to concisely represent the changing conditions of the search space. We use a directed graph $G_t(V, E)$ (Figure 5-3) called the network dynamism graph to represent the flow of candidates and agents between grid-blocks in the search space of dimension (m,n) at time t. We define the

set of vertices $V = V_I \cup \{v_{0,0}\}$ where V_I is the set of nodes representing the grid-blocks and vertex $v_{0,0}$ represents the external environment. Hence $v_{0,0}$ denotes the node from which new candidates enter the search space and to which the fully processed candidates leaving the search space go. Vertex set $V_I = \{v_{i,j}\}$ where $v_{i,j}$ represent the grid-block $B(i,j)$. The edge set $E = E_I \cup E_E$, where E_I is the set of edges between nodes in V_I, represents the flow of candidates and agents due to internal dynamism. Hence, set $E_I = \{e_{i,j}^{k,l}\}$ where $e_{i,j}^{k,l}$ is a directed edge from vertex $v_{i,j}$ to $v_{k,l}$ such that $m \geq k \geq i$, $n \geq l \geq j, i \neq 0, j \neq 0$. On the other hand, the set of edges E_E represents the flow of candidates and agents due to external dynamism.$E_E = \{e_{0,0}^{1,1}\} \cup \{e_{m,j}^{0,0}\}$ where m is the number of columns in the grid search space and $1 \leq j \leq n$.

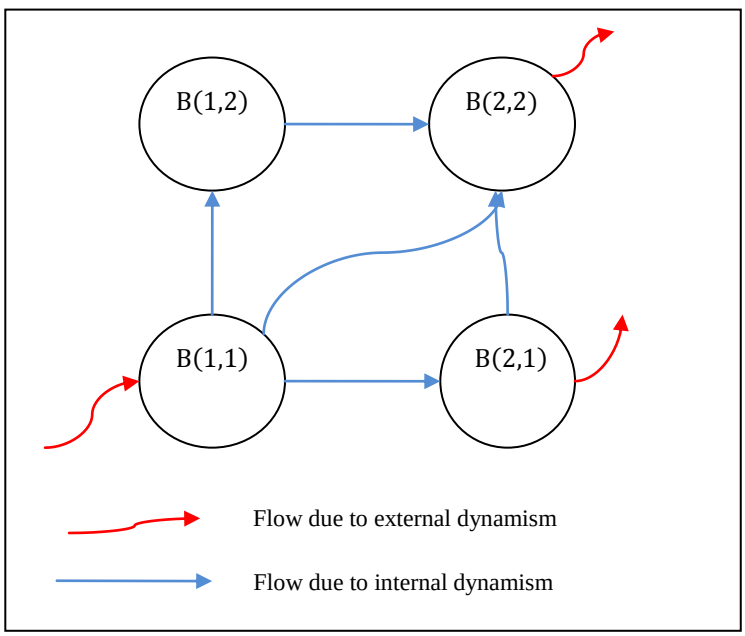

Figure 5-3 Internal dynamism in the search space

Since G_t is a directed graph, each node has a set of in-flow and out-flow edges. Since the generated partial-values of a candidate are monotonically non-decreasing, each vertex $v_{i,j} \in V_I$ has $((i * j) - 1)$ incoming edges and $((m - i + 1) * (n - j + 1) - 1)$ outgoing edges that represent the internal dynamism. It is clear that using common flow metrics such as load, rate of flow, etc. with G_t can help in

qualitative analysis of the dynamism in the search space. One such metric is the load. The load $L(i,j,t)$ in a grid $B(i,j)$ at time t may be defined as:

$$L(i,j,t) = \frac{N_D(i,j,t)}{N_A(i,j,t)}$$

where $N_D(i,j,t)$: No. of candidates in grid-block $B(i,j)$ at time t and $N_A(i,j,t)$: No. of agents in grid-block $B(i,j)$ at time t.

5.4.3 Interaction Module

This module facilitates messaging between agents. Agent identification and addressing are important for sending messages between two agents. Although agent ID is sufficient for messaging, sending messages without an explicit destination address is expensive as it would require flooding or broadcasting to all the agents. However, we are able to avoid such inefficient communication procedures by using the coordinates of the grid-block as the address for agents assigned to it. This simplifies routing of messages. We can also define a distance metric $D_G(a,b)$ between agents using the grid-block coordinates. An example of such a definition of $D_G(a,b)$ between agents a and b which are assigned to home-blocks $H_a(i,j)$ and $H_b(k,l)$ respectively is $D_G(a,b) = \max(|i-k|,|j-l|)$. The distance metric can also be used to qualitatively measure the cost of communication. It also helps us to define a neighborhood (Figure 5-4) of agents, candidates and grid-blocks for a particular agent, candidate or grid-block. For example, an agent a will have home-block $B(k,l)$ as its neighborhood of radius 0, the grid-blocks $\{B(i,j)\}$ where $\max(|i-k|,|j-k|) = 1$, as its neighborhood of radius 1 and so on.

Apart from directed agent-to-agent communications, the agents can also employ short range and long range broadcast communications using the concept of neighborhoods. In the short range communications, the messages are sent within the home-block and in long-range communications, they are sent to other grid-blocks. Agents use asynchronous communication to send information such as the number of

candidates, agents to other grid-blocks. Using this information, the agents update the search space model and recalculate the network metrics.

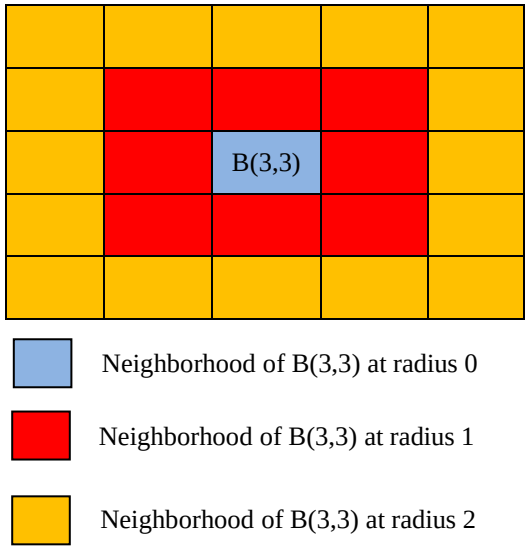

Figure 5-4 Neighborhood in the grid search space model

Our framework is flexible enough to support both synchronous and asynchronous communications. For asynchronous communications, each grid-block can have a buffer in which the agents read/write their messages. Hence sending a message between two agents a and b consist of writing the message in the buffer of the home-block $H_b(x, y)$ of agent b and then b reading from the buffer. In this way, extensive control signaling before each communication step can be avoided.

5.4.4 Prediction Module

The prediction module contains algorithms to approximate the future states of the search space. Flow metrics such as in-flow/out-flow rates and performance metrics such as load $L(i, j, t)$ which are defined in the network dynamism graph change with time. A snapshot of the search space at time t can be represented by the network dynamism graph, G_t. Forecasting algorithms that use flow rates of agents and candidates can be used with G_t to estimate the graph at time $t + \Delta t$, $G_{t+\Delta t}$. This change in the network

can be defined as a set of functions $f_{\Delta t}: G_t \rightarrow G_{t+\Delta t}$ and helps in formally representing the dynamism in the search. We are able to model either the static state at time step t or the dynamic state for a time interval by using different combinations of the framework components.

The static state of the network is depicted by G_t the can be calculated using a combination of search space model and interaction model as shown below.

Search space model + Interaction module => Static snapshot of the search space

The network dynamism graph is updated using the information gathered by the interaction module. In order to model the dynamism over a period of time t to $t + \Delta t$, we use a set of network graphs $G_t, G_{t+1} \ldots . G_{t+\Delta t}$. The network graph at future time steps is calculated by the components given below.

Search space model + Interaction module + Prediction module => Dynamic model of the search space

In a way, the forecasting algorithms provide the functional mapping between network states at different time instants.

The prediction module also incorporates algorithms to predict future partial-values of the candidates. These algorithms may use previous partial-values of the candidates to predict the future values and also assign it a preference or priority value. The priority values represent the potential of the candidate to have a high final similarity value. With the help of prediction module, this framework allows for design of a wide range of candidate selection algorithms by using a mix of global factors (future search space states) and local factors (future values of the candidate).

5.4.5 Internal Model

The internal model component represents the decision making process of the agent. The decisions that the agent has to make while processing the candidates involve: 1) candidate selection, 2) agent migration, and 3) agent communication. The internal model uses the information from the search space model and

interaction module to perform these decision processes in separate sub-components as depicted in Figure 5-5. The three sub-components are:

1. **Agent Migration Sub-component**: This component contains algorithms that control the movement of the agents in the search space. These algorithms calculate the status of the search space from the information in the search space model and use it to determine whether the agent should stay in the current grid-block or move. The quality of results and load conditions in other grid-blocks are relayed to agents through the interaction module. Using its knowledge of the search space and the current results, an agent makes a decision whether to be more explorative and exploitative. Based on this, the agent selects the grid-block to move to. The decision making about agent mobility is represented in the model as follows:

   Search space model + Interaction module + Prediction module + Agent migration sub-component => Agent mobility

   Due to the flexibility and component based architecture of the framework, the agent mobility algorithms can be either simple or complicated. Simple algorithms may use only the static snapshot of the search space and the more complicated ones can incorporate the future search space conditions forecasted by the prediction module.

2. **Agent Communication Sub-component**: This component decides when to communicate and what data to send. The frequency of sending and receiving messages may be constant or linked to the dynamism in the search model. In a large complex environment, each agent has a limited view of the search space. Hence the agents communicate their knowledge of the local neighborhood with each other to build a more complete understanding of the search space. The agent communication sub-component collects this local information and also selects the agents it wants to send to. This sub-component works with the interaction module to send and receive messages. The interaction model is responsible for routing the messages to the appropriate grid, encapsulating the messages, etc. while

the agent communication sub-component decide on the destination of the messages, frequency of the messages, the range of the broadcast messages, etc. It can also take advantage of broadcast communication routines of the interactions module. The communication decision process can be represented in terms of the framework components as:

Search space model + Interaction module + Agent communication sub-component => Agent communication

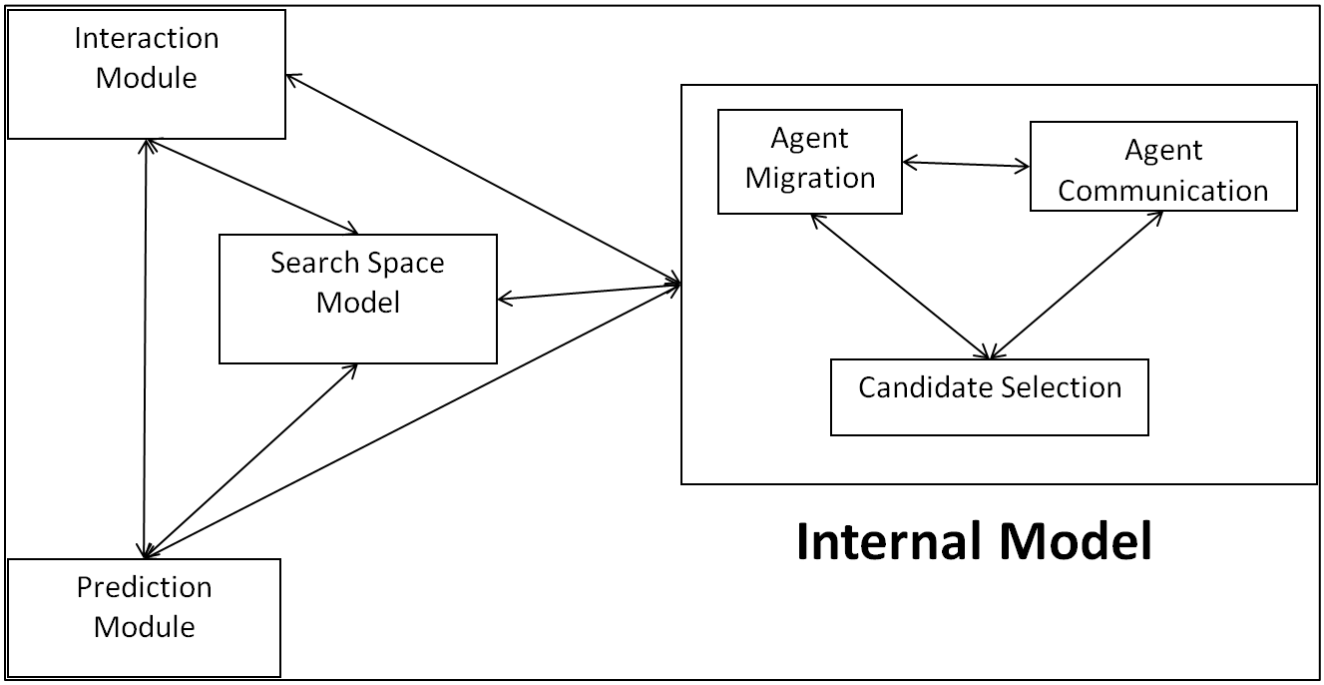

Figure 5-5 Internal Model

3. **Candidate Selection Sub-component**: The methods for selecting candidates are implemented in this sub-component. These algorithms will use the conditions in the search space and the priority metrics of individual documents to pick a candidate for processing. This component also determines the amount of processing time to be applied to the selected candidates.

5.5 Summary

In this chapter, we presented a framework for resource allocation in multiagent system using the partial processing paradigm. The idea behind using the grid search space was laid down and the notion of internal dynamism was discussed. The framework models internal and external dynamism by representing the movement of candidates and agents in the search space using a network graph. Functional mappings between the network snapshots help in modeling the dynamism in the network. The various components of the framework were described and the plug-n-play nature of the framework was emphasized. This component based design helps in easy representation of various agent behaviors and decision making processes.

Chapter 6

Validation of the Resource Allocation Framework

In the previous chapter, we designed a framework for modeling agent resource allocation using the partial-values generated by the partial processing paradigm in large and dynamic search spaces. We now proceed to validate the framework through a set of three simulation experiments complemented by a detailed statistical study. As the initial step in the validation process, we implement the various components of the framework. In the first two experiments, we validate the search space model and demonstrate how the resource allocation process can be modeled within the framework. We compare the performance of resource allocation that uses the grid search space model with other allocation strategies and perform a statistical analysis of the factors that affect their performance. In the final experiment, we use graph-theoretic concepts to model the dynamism in the network, estimate future network states and design agent migration algorithms to achieve better performance. The work presented in this chapter has been published in [**45**].

6.1 Implementation Details

In order to validate the framework, we implement it using carefully designed algorithms and perform simulation experiments to analyze its performance. Agent implementation is done using the JAVA programming language and inter-agent communications are enabled by asynchronous read and write operations into a MySQL database. The object oriented concepts of JAVA helped in easy implementation of the component based architecture of the framework. The simulations are conducted on dual-processor machines with Pentium D 2.80 GHz processors, 4 GB RAM and LINUX operating system. Each experimental run is conducted with multiple agents on a single machine. Depending on the experimental parameters, the number of grid-blocks and the number of agents are varied. In the following sub-sections, we describe the implementation details for each component of the framework.

6.1.1 Search Space Model

The grid based search space model is a key part of the framework and is essential for seamless interoperability of the other components. Recall that we use a grid based search space model in which each grid-block contains candidates with partial-value and process-ratio values within a certain range. Each grid-block is associated with MySQL tables that store the header information of candidates and agents assigned to the grid-block. The header information consists of candidate details such as the total processing time, current process-ratio, current partial-value and agent details such as agent identifier. Though the candidates with similar partial-values and process-ratio may be physically located at different locations in single or multiple databases, they are mapped to their assigned grid-blocks using the MySQl table as depicted in Figure 6-1. Agents use the header information from the MySQL tables to select a suitable candidate for processing. After processing a candidate, the agent assigns it to a grid-block based on the new values of partial-value and process-ratio and updates the header values in the tables of the appropriate grid-block. Each agent contains a data structure to hold the values of the nodes and edges of the network dynamism graph G_t. It may be recalled that the graph $G_t(V,E)$ is used to model the dynamism in the search space at time *t*. The set of nodes *V* represent the grid-blocks and the edges E represent the flow of candidates between the nodes. Since the partial-values are monotonically non-decreasing, the candidates in a grid-block $B(i,j)$ will only move to a grid-block $B(k,l)$ where $i \geq k \geq m$ and $j \geq l \geq n$. Hence, edges in the graph $G_t(V,E)$ are unidirectional with respect to flow of candidates. Due to the unidirectional flow, each node $v_{i,j} \epsilon G_t$ has a set of in-coming and out-going edges represented by $E_{in}(i,j)$ and $E_{out}(i,j)$ respectively. They are formally defined as:

$$E_{in}(i,j) = \begin{cases} \{e_{0,0}^{i,j}\} \cup \{e_{x,y}^{i,j} \epsilon\, E\ \forall\, x,y \mid 1 \leq x \leq i, 1 \leq y \leq j, i \neq x \text{ and } j \neq y\}, & if\ i = 1, j = 1 \\ \{e_{x,y}^{i,j} \epsilon\, E\ \forall\, x,y \mid 1 \leq x \leq i, 1 \leq y \leq j, i \neq x \text{ and } j \neq y\}, & otherwise \end{cases}$$

$$E_{out}(i,j) = \begin{cases} \{e_{i,j}^{0,0}\} \cup \{e_{i,j}^{x,y} \epsilon\, E\ \forall\, x,y \mid i \leq x \leq m, j \leq y \leq n\}, & if\ i = m \\ \{e_{i,j}^{x,y} \epsilon\, E\ \forall\, x,y \mid i \leq x \leq m, j \leq y \leq n\}, & otherwise \end{cases}$$

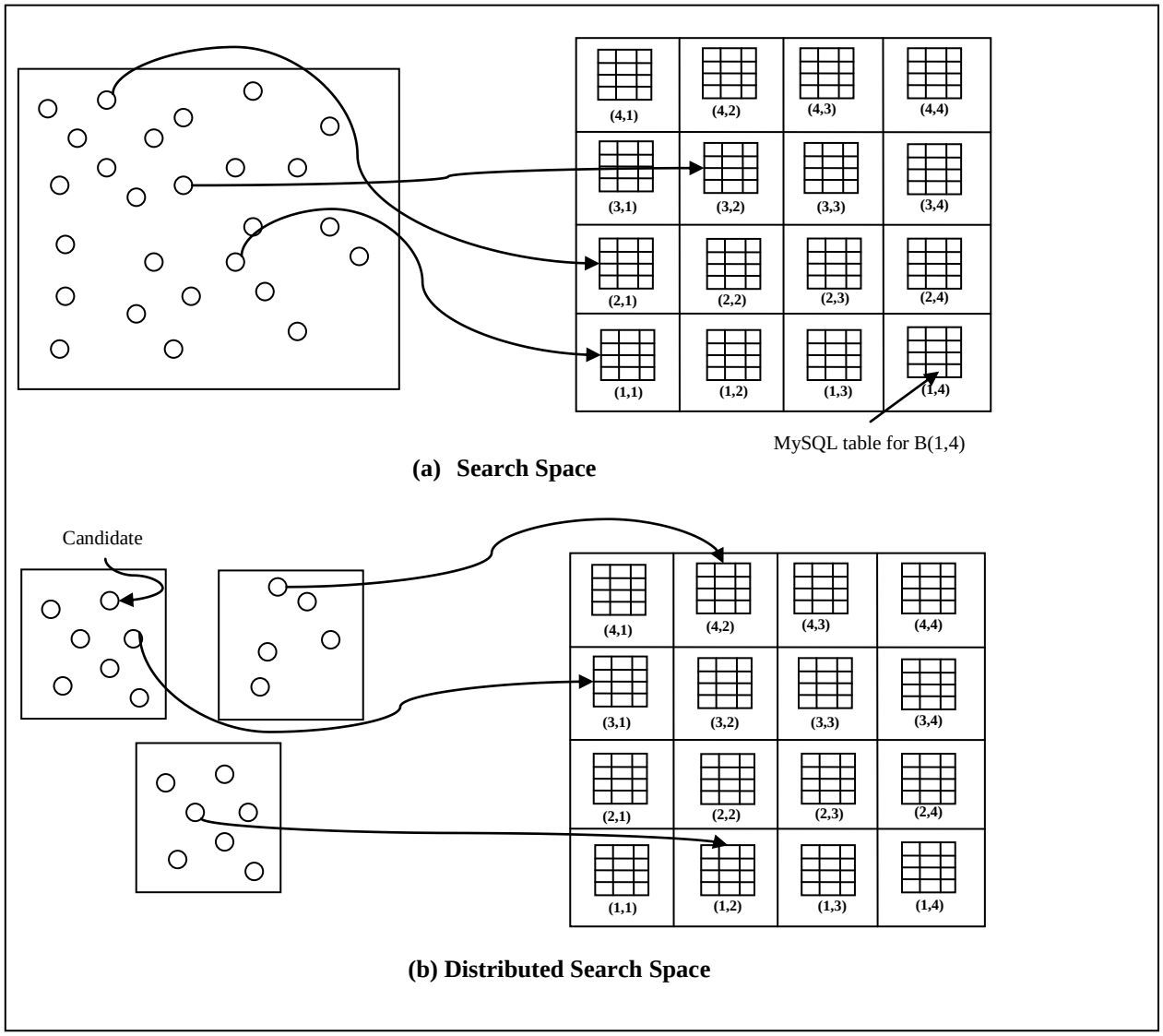

Figure 6-1 Implementation of Search Space

Each edge $e_{i,j}^{k,l} \epsilon E$ has the flow rate $f_{i,j}^{k,l}$ which is defined as the number of candidates passing through the edge per second. Since the flow rate can fluctuate widely over time, we measure a short term or immediate value of the flow rate by using the previous 3 time instants at which candidates have passed through the edge, calculate the intervals between them and find the average interval. In addition to the metrics in the edges, each node $v_{i,j} \epsilon V$ in $G_t(V, E)$ stores values for the following metrics:

1. $g_{in}(i, j)$: in-flow rate of the agents into the grid-block $\boldsymbol{B(i, j)}$

2. $g_{out}\ (i,j)$: out-flow rate of the agents out of the grid-block $\boldsymbol{B(i,j)}$
3. $N_{agent}\ (i,j)$: no. of agents in grid-block $\boldsymbol{B(i,j)}$
4. $N_{candidate}\ (i,j)$: no. of candidates in grid-block $\boldsymbol{B(i,j)}$

6.1.2 Interaction Module

Agents communicate with each other using the interaction module. In our implementation, we avoid flooding of messages by using short range and long range broadcast communications within a neighborhood. In short range broadcast, the messages are sent only to agents in the home grid-block. On the other hand, long range broadcast messages are sent to all agents in grid-blocks that lie within a particular radius. The distance between two agents is calculated by using the D_G metric defined in Section 5.4.3 . The messages sent by agents may be classified into two types based on the information they convey: 1) result, and 2) dynamism. Using this information, the agents make a determination of the search space conditions and the quality of the relevant candidates discovered so far. This will in turn determine the mobility of the agents within the search space. The main types of messages are:

1. **Result Messages**: Due to the design of the grid search space model, the relevant candidates will lie in the set of top-grid-blocks T_B. Depending on the relevant candidates retrieved so far, each agent must make a decision whether it wants to be explorative or exploitative. For example, when the results do not change for a time period, agents become exploitative in order to process candidates that have a higher potential to be relevant so as to quickly produce fresh results. On the other hand, when the result set contains new results, some of the agents may decide to invest resources in explorative regions. Each agent will periodically query the MySQL tables in T_B to get the list of relevant candidates. This list of relevant candidates that have been discovered by time t is called the result-set or $R(t)$ and it changes with addition of newly discovered relevant candidates. Using this information the following metrics, which are utilized in the agent mobility sub-component of the internal model, are calculated:

a. Recall-Ratio ($R_{recall}(t)$): defined as the ratio between the number of relevant candidates discovered by time t to the total number of top candidates. In the experimental setups, we make two simplifications: first, all candidates that have a final value equal to or greater than 0.9 are relevant candidates; second, the total number of relevant candidates are fixed before hand and made known to the agents.

b. Discovery-Time ($T_{disc}(i)$): of a candidate i is the time instant at which it is found to be relevant or in other words, the time instant when its partial-value becomes equal to or greater than 0.9.

c. Result-Age ($R_{age}(t)$): is the period of time for which the current result-set has remained unchanged.

2. **Dynamism Messages**: Dynamism messages inform agents about the movement of candidates and agents in the search space. When an agent enters or leaves a grid-block, it leaves a message in the database tables of the source and destination grid-blocks indicating its departure and arrival respectively along with the timestamp. Similarly, when a candidate is assigned to a new grid-block after being processed by an agent, the agent sends a message to the agents in both its home-grid and the new grid-block of the candidate indicating the transfer. Besides these messages, the agents also get information about the number of candidates and agents in the neighborhood. In order to keep the communication costs low, we restrict the neighborhood to a radius of 1. Since the agents are likely to have current information about the neighborhood, we restrict agent migration to only the grid-blocks in the neighborhood. Note that these restrictions in agent movement and communication are aspects of our implementation and are not limitations imposed by the framework. The agents also send information regarding the flow rates of the agents and candidates from the home-grid to other agents in the neighborhood. The agents use the dynamism messages to update the network graph G_t. Using the information in G_t, the load metric is calculated. Load ($L_{B(x,y)}(t)$) for a grid-block $B(x, y)$ at time t is defined as the ratio between number of candidates and the number of agents in the grid-block.

Values of the number of agents and candidates are calculated from the header information stored in the MySQL tables corresponding to the grid-block. Agents also calculate the load conditions in the neighborhood.

6.1.3 Internal Model

Internal model encapsulates the decision making process of the agent. The algorithms use the information in the search space model, in order to make decisions regarding agent communication, candidate selection and agent migration.

Agent Communication: Agents share information regarding dynamism and search results, with each other, using the interaction module. Since message passing is implemented by reading and writing into the database tables of the grid-blocks, this sub-component has routines that decide when and where to read/write messages. Some message passing is triggered by events such as departure of the agent to a new home-block or when a candidate is moved to a different grid-block. Other message passing is done periodically to gather information regarding the number of candidates and agents in the home-grid and in the neighborhood.

Candidate Selection: The candidate selection procedure adopted here is similar to the one adopted for text processing and image processing algorithms. Each candidate has a priority metric that measures its potential to be relevant. We employ the equation for priority (as given in Section 2.5.3) with a modification - the future partial-value of the candidate is predicted using a linear regression [**189**] based algorithm implemented in the prediction component. Based on candidates' priority values, each agent finds the top 10 candidates using the header information stored in the database table of the home-block and then randomly selects one for processing.

Agent Mobility: The algorithms in the agent mobility sub-component periodically calculates the result metrics $R_{avg}(t)$ and $R_{age}(t)$. Based on the values of these metrics, the explorative/exploitative behavior of the agent is changed. The decision making concerning this change in behavior is done with the help of

Bayesian Knowledge Bases (BKB) [**190**]. Bayesian Knowledge Base provides a probabilistic framework for modeling decision making with imprecise and uncertain knowledge. It has an advantage over traditional Bayesian Networks [**191**] because of its ability to represent scenarios without having to explicitly model all possible instantiations or states of the random variables. This makes BKBs suitable for real world scenarios with large number of random variables. BKBs also allow for easy encoding of IF-THEN rules using the principles of conditional probability. BKB is a directed graph with two types of nodes: I-nodes and S-nodes. The edges represent the conditional dependencies between the nodes. I-nodes are used to represent the various instantiations of the random variables and the S-nodes denote quantitative representation of the conditional dependencies between the I-nodes. A sample BKB is depicted in Figure 6-2 where the rectangular nodes represent the I-nodes and the circular nodes represent the S-nodes. In each I-node, the horizontal label is the random variable it represents and the vertical label is the instantiation of the random variable. Each S-node has a value corresponding to the conditional probability it represents.

The random variables (r.v.) and their instantiations used in the agent mobility decision making are listed in Table 6-1. The input variables are RESULT_AGE and RESULT_AVERAGE and the output variable is RESULT_OUTPUT. The r.v. RESULT_AGE has three instantiations NEW, OLD and OLDER, each of which corresponds to a particular range of values of the metric $R_{age}(t)$. The ranges for these variables are based on the length of the candidates in the search space. For example, if $R_{age}(t)$ is greater than the average time to process 10 candidates, then the r.v. RESULT_AGE is set to OLDER. Similarly, the r.v. RESULT_AVERAGE has three instantiations HIGH, MEDIUM and LOW, each corresponding to a range of values of the metric $R_{avg}(t)$. The output variable RESULT_OUTPUT has three instantiations EXPLORE, EXPLOIT and MEDIUM which is mapped to value ranges of the risk parameter R_{risk} and thus represents the explorative/exploitative nature of the agents. In addition to EXPLORE and EXPLOIT states, we use MEDIUM state to represent agent behavior that has a mix of EXPLORE and EXPLOIT behaviors. The decision making in the BKBs proceeds as follows. The values of the metrics $R_{age}(t)$ and

$R_{avg}(t)$ are calculated using the information gathered from the top-grid-blocks T_B. Depending on the value of $R_{age}(t)$, an instantiation of RESULT_AGE is set to true while its other instantiations are set to false. This process, called "setting the evidence", is also carried out for RESULT_AVERAGE. Using the evidence and the dependency relationships between the nodes in BKB, we calculate the probabilities for r.v. RESULT_OUTPUT to be EXPLORE, EXPLOIT or MEDIUM. Recall that each grid-block $B(x, y)$ has a risk parameter $R_{risk}(x, y)$ (described in Section 5.4.2). We add the normalized value of the risk parameter (multiplied by the probabilities calculated using the BKB) with the normalized load metric $L_{B(x,y)}$ for each neighboring grid-block. A probabilistic selection procedure, in which the probability that a grid-block is selected is proportional to the normalized sum of load and risk metrics, is used to select a grid-block that the agent migrates to. A more sophisticated agent mobility decision making that uses approximations of the future values of the load in the neighborhood can also be designed. Load prediction algorithms in the prediction module are used to calculate the future load values. The selection procedure will then use normalized values of current and future load values of the neighborhood grid-blocks along with the risk parameter to select a grid-block to move to.

Random Variable	**Instantiation States**	**Metric**
RESULT_AGE	NEW	Result-Age $R_{age}(t)$
	OLD	
	OLDER	
RESULT_AVERAGE	HIGH	Recall-Ratio $R_{recall}(t)$
	MEDIUM	
	LOW	
RESULT_OUTPUT	EXPLORE	N/A
	MEDIUM	
	EXPLOIT	

Table 6-1 Random variables and instantiations used in BKB for agent mobility decision making

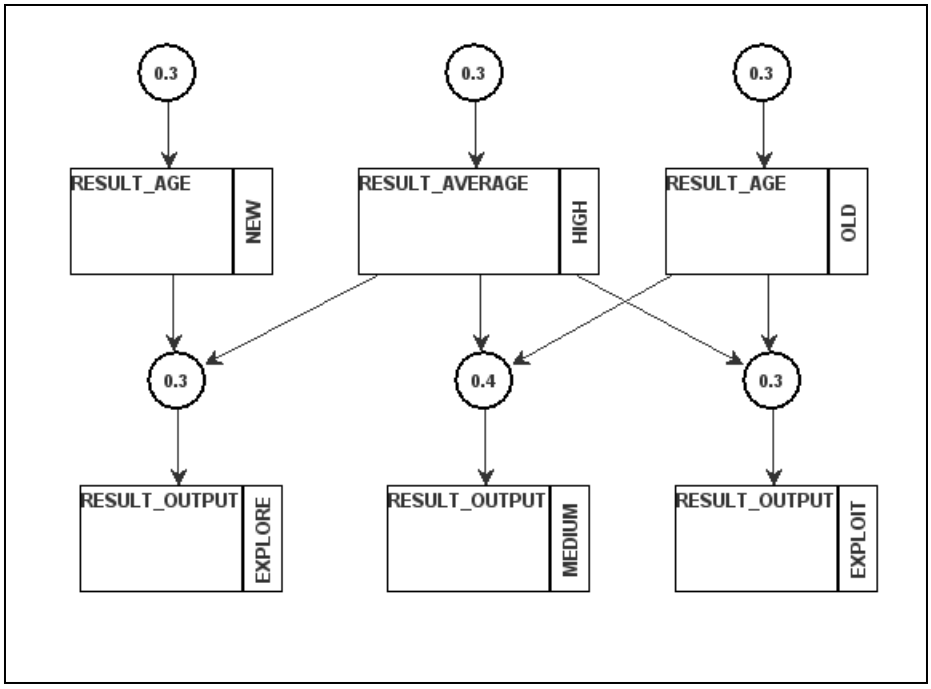

Figure 6-2 Portion of the agent mobility rule-set in BKB representation

6.1.4 Prediction Module

The methods implemented in the internal model use the algorithms in the prediction module to forecast future conditions of the search space. Since the decision makings are for agent migration and candidate processing, we implement algorithms for forecasting the future values of search space metrics such as load and the future values of partial-values of candidates.

Candidate Prediction

We implement a prediction algorithm based on linear regression [**189**] to determine the future partial-value of the candidate for a given process-ratio. The future partial-values of the candidates are used by the candidate selection sub-component to calculate the priority values of the candidates. The previous history of each candidate that is processed by the agent is used as the training data to calculate the regression parameters. The history consists of a sequence of partial-values and its corresponding process-ratios. The training sample is refreshed as new candidates are processed and the regression parameters are recalculated.

Load Prediction

We use flow rates of agents and candidates to estimate the number of agents and candidates in a grid-block at some future time $t + \Delta t$ using the following formulae.

$$N_{agent}\ (i, j,\ \Delta\, t) = N_{agent}\ (i, j) + \left(g_{in}(i, j) - g_{out}\,(i, j)\right) *\ \Delta\, t$$

$$N_{candidate}\ \ (i, j,\ \Delta\, t) = N_{candidate}\ \ (i, j) + \left(\sum_{e_{x,y}^{i,j} \epsilon E_{in}\,(i,j)} f_{x,y}^{i,j} - \sum_{e_{i,j}^{x,y} \epsilon E_{out}\ (i,j)} f_{i,j}^{x,y} \right) * \Delta t$$

where

$N_{agent}\ (i, j, \Delta t)$: *Approximate no. of agents in grid-block* $B(i, j)$ *after* Δt *time interval*
$N_{candidate}\ \ (i, j, \Delta t)$: *Approximate no. of candidates in grid-block* $B(i, j)$ *after* Δt *time interval*

The flow rates for a grid-block are calculated by using the average intervals at which the agents and candidates enter or leave the grid-block. These values are stored in the network graph and are updated using the information in messages received from other agents and by querying the database of the home-grid. Since load is the ratio of number of candidates and agents, we can also calculate future load values. Since these flow rates formulae are coarse representation of the changes in the search space, this methodology can be used to do only short term forecasting. Designing a more robust algorithm that models the stochastic nature of the search space more closely is part of the future work.

6.2 Experimental Testbed

We conduct our experimental simulations on large testbeds of synthetic candidates. A synthetic candidate consists of a sequence of randomly generated 2-tuple $\{(r_d(t_1), v_d(t_1)), (r_d(t_2), v_d(t_2)), \ldots\ldots (r_d(t_n), v_d(t_n))\}$ such that $r_d(t_i) < r_d(t_j)$ and $v_d(t_i) \leq v_d(t_j)$ where $t_i < t_j$. $r_d(t)$ and $v_d(t)$ are the process-ratio and partial-value respectively, of the candidate *d* after *t* amount of processing time has been applied. Synthetic candidates are used in place of real documents because it allows the flexibility for easily changing the search space properties such as candidate length (short, long), rate of change in the partial-values, proportion of candidates based on size,

etc. Each candidate has an entry-time at which it enters the search space. By manipulating the entry-time of the candidates, the rate of entry of the candidates can be varied and various degrees of external dynamism simulated. Since we are validating a real time retrieval framework, our aim is to minimize the time it takes to retrieve or discover the relevant candidates. For the experiments, we assume that all candidates that have partial-value greater than or equal to 0.9 is said to be a relevant candidate.

6.3 Experiment-1: Validation of the Search Space Model

In Experiment-1, we focus on validating the fundamental design characteristics of the search space model such as grouping the candidates based on their process-ratio and partial-value and labeling these groups as exploitative or explorative regions. We will show that methods based on this grid-based division of the search space can lead to efficient and dynamic resource allocation. In order to validate the search space model, we construct a multiagent control system called the Mobile-Agent system in which the agents move between grid-blocks as they change their search behavior from exploitative to explorative and vice-versa. We compare its performance with other control systems called the Baseline and Static-Agent system which use resource allocation rules based on simpler search space models.

6.3.1 Control Systems

Each control system is constructed within our framework by implementing only those components that are required for its behavior. The differences between the systems are depicted in Figure 6-3(a-c) where the components that are not used are either shaded or struck through.

Baseline System: In the Baseline system, the search space is monolithic - there is no grouping of similar candidates based on some criteria. All the agents have access to all the documents in the search space at all times. The agents follow a greedy procedure by strictly choosing the documents with the highest partial-values for processing. This control system represents the traditional parallel and distributed methodology of processing candidates. From Figure 6-3(a), we see that the Baseline system is implemented by using only the candidate selection component and candidate prediction algorithm.

Static-Agent System: In this control system, the documents are grouped together on the basis of the partial-value and process-ratio, using grids. However each grid-block is assigned constant resources or agents throughout the duration of the processing. In other words, each agent has a permanently assigned grid-block. Static-Agent system is more scalable compared to the Baseline system because it does a search space partition and has to only sort and select candidates within its home-block. Figure 6-3(b) shows the components of the framework that are used for the Static-Agent system. Recall that sharing information through agent interaction help in agent mobility. Since the agents are immobile in the static system, interactions are not useful. Hence the interaction component and the agent communication sub-component of the internal model are not used. The agent mobility sub-component is also kept inactive. Although the grid space model is used, the network dynamism graph is not implemented as the Static-Agent system does not model movement of agents and candidates between grid-blocks.

Mobile-Agent System: Mobile-Agent system uses all the components of the framework, and it is used to validate our notions of dynamism and search space partitioning by comparing it with the other systems. The grid space model is used, and unlike the Static-Agent system, the agents are free to move between blocks. Agents can process only the candidates in its home-grid. The mobility of the agents is decided based on the conditions in its home-block and its neighboring grid-blocks. We adopt the following restrictions in order to simplify the implementation.

1. Agent can only move from its home-block to its immediate neighborhood. Its immediate neighborhood consists of all the grid-blocks that are at a distance of 1 unit as defined by the distance metric, D_G. This ensures that an agent needs to model only the dynamism in the immediate neighborhood.
2. Only communications with agents in the home-block and the immediate neighborhood is allowed. This limits the communication costs incurred by the agents while gathering information to update the network dynamism graph.

3. Since this is a preliminary validation process, only a simple dynamism model using the load conditions in the local search space is used. A more detailed dynamism model that uses the flow rates of candidates and agents to forecast the future network states is the focus in Experiment-3.

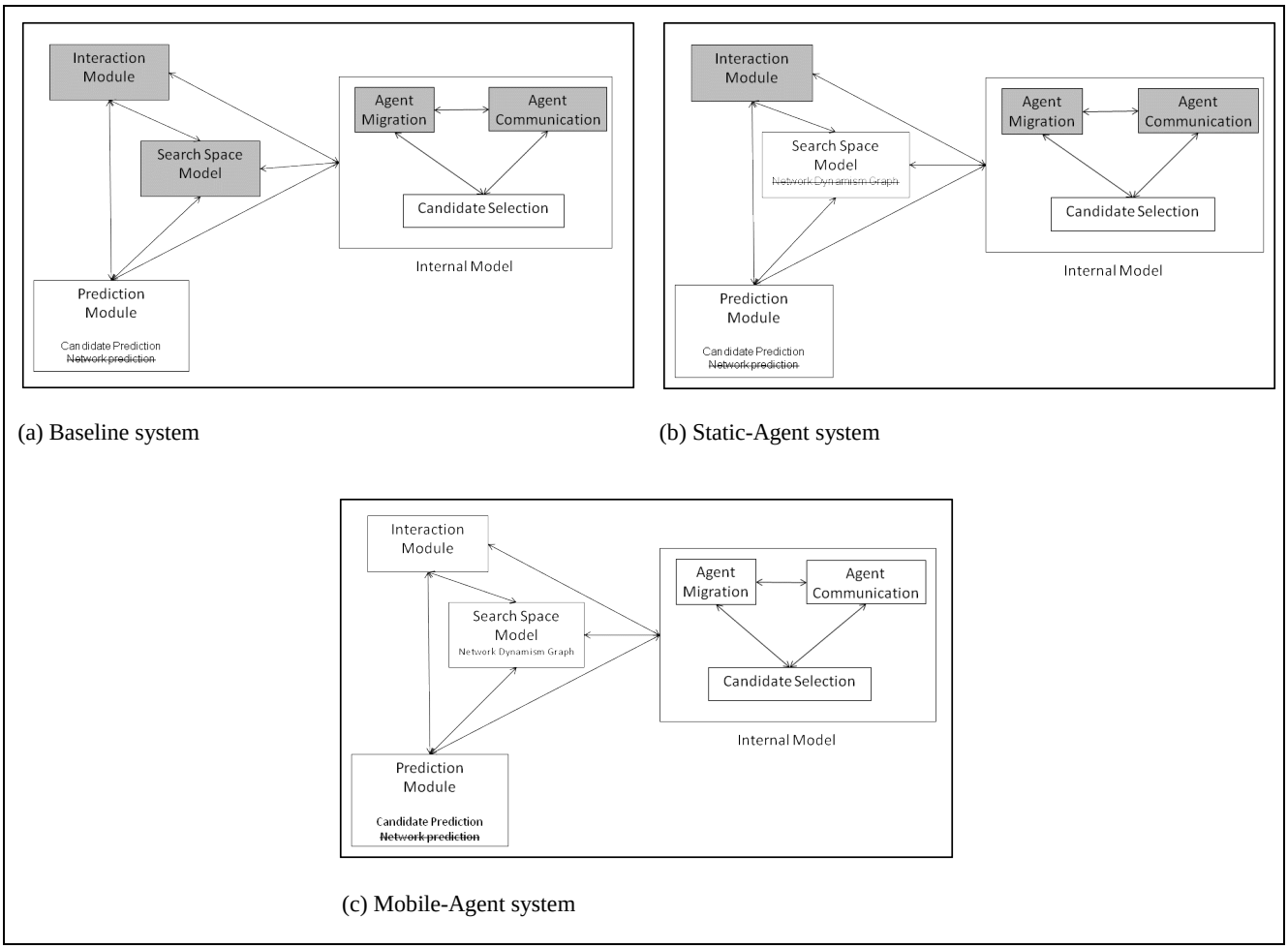

Figure 6-3 Control Sytems for Experiments 1 and 2

It may be noted that our goal here is not to design the optimal communication or prediction algorithms. Our focus is to validate important aspects of the framework and for that purpose; we seek to design efficient communication or prediction algorithms. The validations points are the importance of modeling dynamism, plug-and-play nature of the framework and modeling explorative/exploitative behavior.

6.3.2 Results

For Experiment-1, we built a testbed of candidates by generating sequences of partial-values and process-ratios. The performance metric used to compare system performance is the total-recall time. Total-recall time is the time taken for the system to retrieve all the relevant candidates. Since real time search systems deal with getting the results quickly, total-recall time is an important measure to study its performance. The testbed consists of 1000 candidates with each candidate having an entry-time. The entry-time is the interval of time after the start of simulation that the candidate enters the search space. The candidates are inserted at an average of 10 candidates/s. For each candidate, we assign a value called the total-processing time which is the time required to completely process the candidate. The total-processing time (in seconds) for candidates in the testbed is uniformly distributed in [1, 10]. Each control system is run on the testbed with agents ranging from 8 to 64 and the grid sizes (2, 2) and (4, 4). The experiments are conducted using the randomized block paradigm [**192**] where a block run consists of running the three systems in a randomized sequence. We perform 3 block runs for each combination of agent number and grid-size. The results are collected for each run, and the average and standard deviation of the total-recall times are calculated (Table 6-2). We perform Analysis of Variance (ANOVA) [**192**] on the results to ascertain whether the difference in performance of the control systems is statistically significant.

A study of Table 6-2 and bar charts in Figure 6-4 and Figure 6-6 shows that Mobile-Agent system performs better than the Static-Agent and Baseline systems in all the experimental runs. Baseline system performs the worst as the recall time actually increases with increase in the number of agents. This is due to the contention between the agents as all agents in the Baseline system try to access the candidate with the highest priority. In the Static-Agent system, although the total-recall time decreases with the increase in the number of agents, it decreases at a slower rate than the Mobile-Agent system. The primary reason for the better performance of the Mobile-Agent versus the Static-Agent is that the former is able to minimize agent idleness by moving between the grid-blocks. Although the agents in Static-Agent do not perform expensive communications operations to interact with each other, the cost savings are not enough

to offset the cost due to idleness. The superior performance of the Mobile-Agents system shows that a grid based search space model that categorizes the candidates using their process-ratio and partial-values is indeed viable. Since resource allocation in mobile agents system uses the risk parameter, the experimental results also validates the notion of leveraging exploration and exploitation regions for better performance.

No. of Agents	**Grid Size**	**Total-Recall Time (s)**					
		Baseline		**Static-Agent**		**Mobile-Agent**	
		Avg	Stdev	Avg	Stdev	Avg	Stdev
8	2,2	2863.74	11.18	1284.11	2.98	829.87	6.85
16	2,2	2885.86	6.55	663.34	0.06	432.19	19.20
32	2,2	2992.45	8.60	380.12	2.96	295.93	8.11
16	4,4	2935.13	10.45	1466.79	2.18	698.55	54.05
32	4,4	3000.32	9.32	760.12	2.89	280.24	15.48
64	4,4	3230.09	112.90	417.75	2.82	230.21	4.06

Table 6-2 Comparison of Total-Recall time in Experiment-1

Analysis of Variance or ANOVA [**192**] is a commonly used statistical method for determining whether the mean of two or more population sets are the same or not. We conduct ANOVA analysis on the results of the Experiment-1 to determine whether the superior performance of the Mobile-Agent system is statistically significant. We show a sample of the ANOVA analysis for experimental runs with: 1) grid size = (2, 2), number of agents = 8 (Table 6-3, Table 6-4 and Figure 6-5), and 2) grid size = (4, 4), number of agents = 64 (Table 6-5, Table 6-6 and Figure 6-7). From the graphical ANOVA figures for these runs (Figure 6-5 and Figure 6-7), it is clear that the block deviations and residual values are much smaller than the deviations between the system performances. From the results of the F-test ratio (Table 6-4, Table 6-6), we see that the F-ratio > F_{crit} which leads to the rejection of the null hypothesis - the mean of the population of the performance results of the three systems are the same.

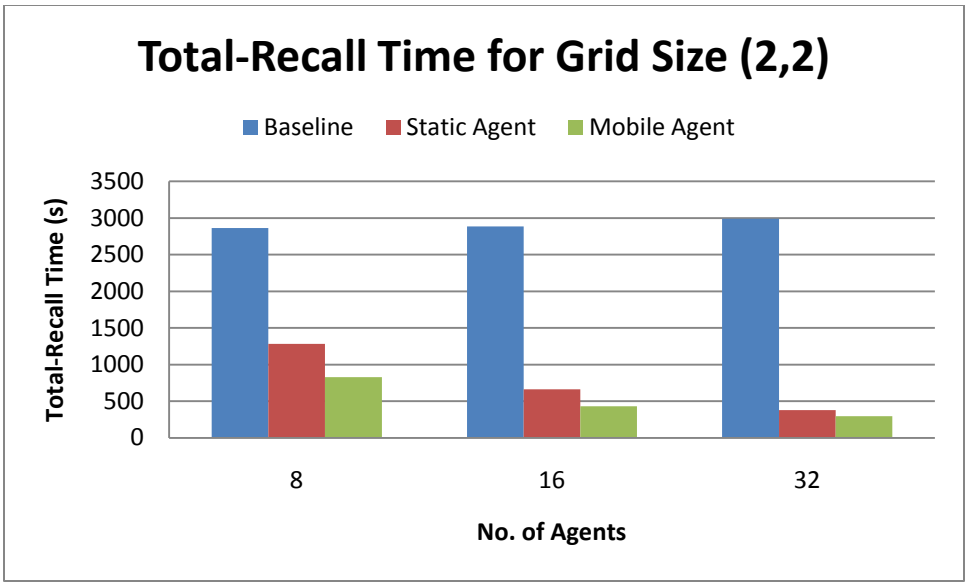

Figure 6-4 Total-Recall time for Experiment-1 with grid size=(2,2)

	Baseline	Static	Mobile	Block Mean	Block Deviation			Method Deviation			Residuals		
Block1	2876.65	1285.78	837.11	1666.51	7.27	7.27	7.27	1204.5	-375.13	-829.37	5.64	-5.60	-0.03
Block2	2857.37	1280.67	829.02	1655.69	-3.55	-3.55	-3.55	1204.5	-375.13	-829.37	-2.82	0.11	2.70
Block3	2857.2	1285.88	823.48	1655.52	-3.72	-3.72	-3.72	1204.5	-375.13	-829.37	-2.82	5.49	-2.67
Average	2863.74	1284.11	829.87										
Grand Average	1659.24												

Table 6-3 Calculation of block deviations, method deviations and residuals for Experiment-1 with no. of agents=8 and grid size=2,2

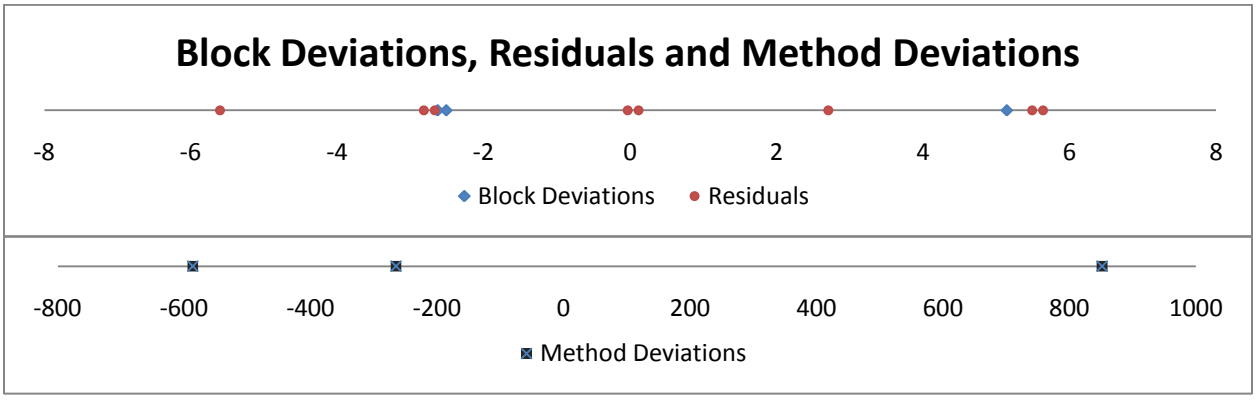

Figure 6-5 Graphical ANOVA for Experiment-1 with no. of agents=8 and grid size=(2,2)

Source of Variation	SS	df	MS	F-ratio	P-value	F-crit
Between Groups	6838192	2	3419096	56710.17	1.48E-13	5.14
Within Groups	361.74	6	60.29			
Total	6838554	8				

Table 6-4 F-test in the ANOVA analysis for Experiment-1 with no. of agents=8 and grid size=2,2

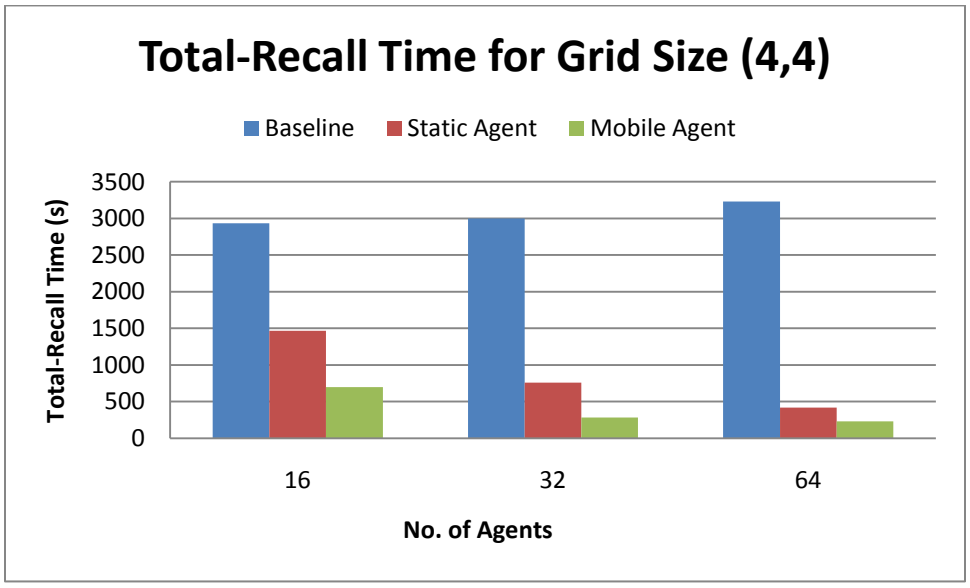

Figure 6-6 Total-Recall time for Experiment-1 with grid size=(4,4)

	Baseline	Static	Mobile	Block Mean	Block Deviation			Method Deviation			Residuals		
Block1	3135.43	419.58	234.38	1263.13	-29.55	-29.55	-29.55	1937.40	-874.93	-1062.47	-65.10	31.38	33.72
Block2	3355.04	414.51	226.27	1331.94	39.26	39.26	39.26	1937.40	-874.93	-1062.47	85.70	-42.50	-43.20
Block3	3199.79	419.17	229.98	1282.98	-9.70	-9.70	-9.70	1937.40	-874.93	-1062.47	-20.59	11.12	9.47
Average	3230.08	417.75	230.21										
Grand Average	1292.683												

Table 6-5 Calculation of block deviations, method deviations and residuals for Experiment-1 with no. of agents=64 and grid size=4,4

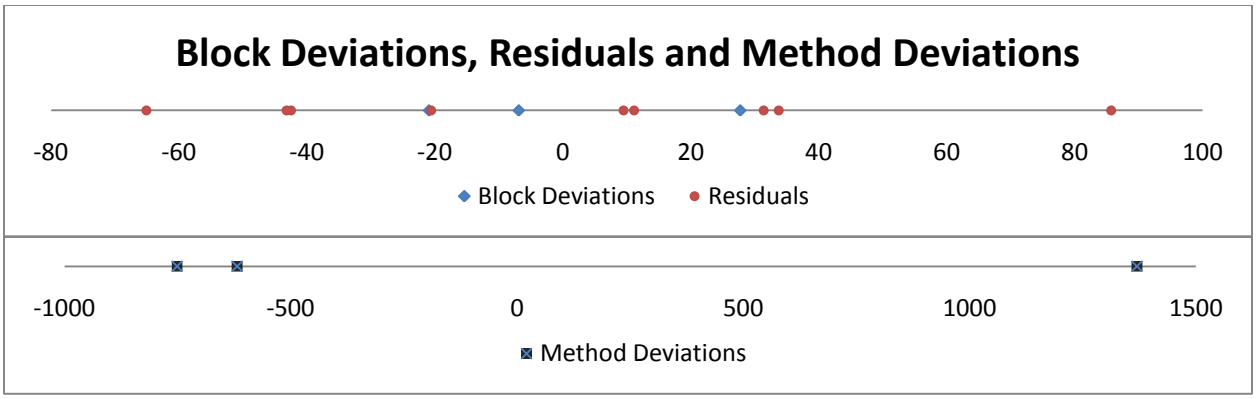

Figure 6-7 Graphical ANOVA for Experiment-1 with no. of agents = 64 and grid size = 4,4

Source of Variation	SS	df	MS	F-ratio	P-value	F-crit
Between Groups	16943651	2	8471826	1990.25	3.41E-09	5.14
Within Groups	25539.94	6	4256.66			
Total	16969191	8				

Table 6-6 F-test in the ANOVA analysis for Experiment-1 with no. of agents = 64 and grid size = 4,4

6.4 Experiment-2: Factorial Analysis

Experiment-1 was only a preliminary validation of the search space model. It demonstrated that using the notion of exploration/exploitation and modeling the load conditions in the neighborhood lead to efficient allocation of resources. In Experiment-2, we go in for detailed analysis of the effects of various search space variables on the system performance using a 2-level fractional factorial analysis method.

6.4.1 2-Level Factorial Analysis

In Experiment-2, we perform a statistical analysis of the control systems using factorial design methodology [**192**] [**193**]. As in Experiment-1 total-recall time is used as the performance metric in these experiments. Factorial designs are effective when the number of factors to be studied is large and/or the factors vary over a wide range. A factor is an independent variable that is known to affect the system performance. In a 2-level factorial analysis, the performance of the control systems is studied at two

levels of each factor: 1) Min-level which is a small value, and 2) Max-level which is a large value. By analyzing the change in the output variable with variations in factors, the effects of the factors are calculated. An effect is defined as a change in the output/response variable due to changes in the factors. The effects are categorized as main effects and interactions. Main effect is caused by a factor without any contribution from the other factors. Effects due to two or more factors acting in tandem are called interactions. Thus effect of two factors taken together is called two-factor interaction; effect of three factors is called three-factor interaction and so on. In a full-factorial study, the control systems are run at every combination of the values of the factors and the effects calculated. The levels of the factors for each run in a 2^3 full factorial design are depicted in Table 6-7. In this table, we see that the min and max levels of each factor are represented as -1 and +1 respectively. The values and variations of these factor values can also be represented geometrically using a cube (Figure 6-8) where each vertex denotes the values of the output variable at various combinations. The formulae for the main effects and higher-order interactions are given in Equation 6-1, Equation 6-2 and Equation 6-3.

$$main\ effect\ of\ x_1 = \frac{y_2 + y_4 + y_6 + y_8}{4} - \frac{y_1 + y_3 + y_5 + y_7}{4}$$

$$main\ effect\ of\ x_2 = \frac{y_3 + y_4 + y_7 + y_8}{4} - \frac{y_1 + y_2 + y_5 + y_6}{4}$$

$$main\ effect\ of\ x_3 = \frac{y_5 + y_6 + y_7 + y_8}{4} - \frac{y_1 + y_2 + y_3 + y_4}{4}$$

Equation 6-1 Main effects

$$2-factor\ interactions\ of\ x_1x_2 = \frac{y_1+y_4+y_5+y_8}{4} - \frac{y_2+y_3+y_6+y_7}{4}$$

$$2-factor\ interactions\ of\ x_1x_3 = \frac{y_1+y_3+y_6+y_8}{4} - \frac{y_2+y_4+y_5+y_7}{4}$$

$$2-factor\ interactions\ of\ x_2x_3 = \frac{y_1+y_2+y_7+y_8}{4} - \frac{y_3+y_4+y_5+y_6}{4}$$

Equation 6-2 2-Factor interactions

$$3-factor\ interactions\ of\ x_1x_2x_3 = \frac{y_2+y_3+y_5+y_8}{4} - \frac{y_1+y_4+y_6+y_7}{4}$$

Equation 6-3 3-Factor interaction

run	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	**y**
1	-1	-1	-1	y_1
2	-1	-1	+1	y_2
3	-1	+1	-1	y_3
4	-1	+1	+1	y_4
5	+1	-1	-1	y_5
6	+1	-1	+1	y_6
7	+1	+1	-1	y_7
8	+1	+1	+1	y_8

Table 6-7 Full factorial design for 3 factors

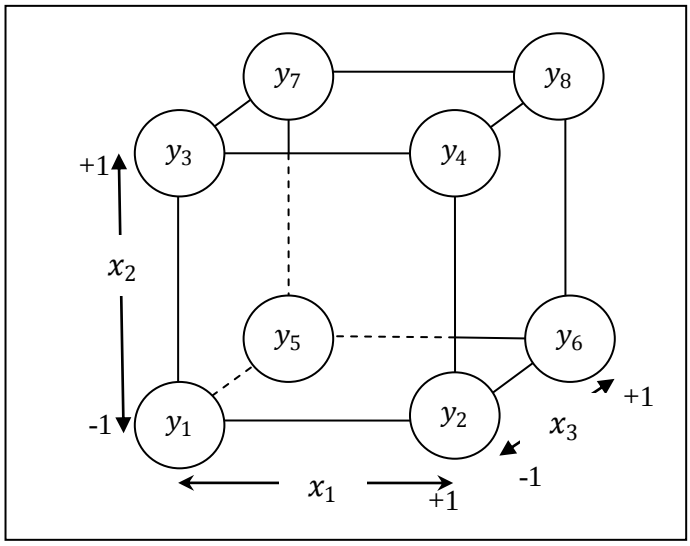

Figure 6-8 Geometric representation of a 3-factor full factorial design

For a full factorial study of the Baseline (Mobile-Agent and Static-Agent) systems with 5(6) factors, each control system is run with $2^5(2^6)$ combinations of the factors. Since we are interested only in the main effects, we use a fractional factorial design that requires only a fraction of the runs employed in a full-factorial design. In the next sub-section, we identify the factors that we consider important in large and dynamic spaces and perform a fractional factorial analysis on these factors to determine their effects on the performance of the control systems.

6.4.2 Factors

The factors used in the study can be divided into two types. Some of them such as search space size, candidate processing time, etc. are properties of the search space and others like number of agents and grid size are properties of the multiagent system. The factors used in the factorial analysis are listed in Table 6-8 and are described below.

1. **Similarity Distribution (S_f)**: The number of non-relevant candidates in the search space that have high similarity final values can have an important bearing on the performance of resource allocation algorithms. Intuitively, we can see that picking the relevant candidates in a search space with large

number of non-relevant candidates with high final values is harder than in a search space where most non-relevant candidates have low final values. Since the average final similarity values in such search spaces are high, large resources might be diverted to process non-relevant candidates and wasted. In the experimental testbed, we define the similarity distribution factor as the percentage of candidates that have similarity values in the range [0.5-0.9]. The values of the two levels for this factor are set at 0.2 and 0.8.

2. **No. of Agents (M_f)**: The agents constitute the computing resources applied and as such are an important factor in determining the performance. After conducting preliminary experiments, it was decided to use the values 16 and 64 as the minimum and maximum levels of this factor.
3. **Grid Size (G_f)**: is a 2-tuple (m,n) representing the number of grid-blocks along the X-axis and Y-axis respectively. The number of grid-blocks in the search space model determines the granularity of the grouping of the candidates. The two levels used for this factor are (2,2) and (4,4).
4. **Processing Time (T_f)**: is the time taken to fully process a candidate. This may also be considered as the size of the candidate. Large non-relevant candidates that appear promising to the candidate selection algorithms may block resources from being allocated to more deserving ones. On the other hand, the trend of the partial-values may be quickly detected and correctly identified with large candidates. We study this tradeoff in this experiment by setting the levels of this factor to 10s and 50s.
5. **Entry Rate (R_f)**: is the rate at which candidates enter the search space. By using this factor, we study the effects of external dynamism on system performance. The two levels of this factor are 1 candidate/s and 10 candidates/s.
6. **Search Space Size (N_f)**: is the number of candidates in the search space. The two levels of this factor are 200 and 1000.

Factors		Min Level (-1)	Max Level (+1)
Symbol	**Description**		
S_f	Similarity Distribution	0.2	0.8
M_f	No. of Agents	16	64
G_f	Grid Size (m,n)	2,2	4,4
T_f	Processing Time	10s	50s
R_f	Entry Rate	1 candidate/s	10 candidates/s
N_f	Search Space Size	200	1000

Table 6-8 Factors used in the factorial analysis for Experiment-2

6.4.3 Fractional Factorial Analysis

We have selected 5 factors for Baseline system (Table 6-15) and 6 factors for both Static-Agent (Table 6-9) and Mobile-Agent (Table 6-12) systems. In order to do a full factorial analysis with 6 (5) factors, we need 2^6 (2^5) runs. Each run is also repeated multiple times to study the variance. Since the focus of our analysis is to study only the main effects, it is not necessary to perform the tedious full factorial study. It is possible to conduct detailed analysis of the main effects of the parameters using a fractional factorial design with 16 runs. In order to accommodate 5 (for Baseline) or 6 (for Static-Agent and Mobile-Agent) factors within the fractional design, we use the principle of aliasing or confounding [**192**]. For example, consider the fractional design of the Mobile-Agent and Static-Agent systems. For aliasing, we represent the factors using their dummy variable notations as given in Table 6-9 (Static-Agent) and Table 6-12 (Mobile-Agent). The aliasing is done using the generating and relations listed in Equation 6-4. The identity relations (Equation 6-4 (iv) – (vi)) are calculated by multiplying the generating relations in Equation 6-4(i), Equation 6-4(ii) and Equation 6-4(iii) by d, e and f respectively. Since the largest term in the identity relationships have 4 variables, this factorial design is said to be of resolution 4. Specifically, this design is denoted as 2_{IV}^{6-3} where the subscript "IV" denotes the resolution of the design and the superscript "6-3" indicates that the design has 6 factors and the number of generating relations required for aliasing is 3. Due to aliasing, the main effects of the factors are biased by the effects of other factors. This confounding pattern for each dummy variable is calculated by multiplying each identity relationship by the dummy variable. The confounding patterns are listed in Equation 6-5(i)-(vi). For example, in

Equation 6-5(i), we see that the effect of the variable *a* is confounded by the second-order interactions of *be*, *cf* and third-order interactions of *bcd*. We ignore the effects of third and higher-order interactions. By removing these interactions, we establish the aliasing relations in Equation 6-5(vii)-(xii).

$d = abc$	(i)	=>	$I = abcd$	(iv)
$e = ab$	(ii)	=>	$I = abe$	(v)
$f = ac$	(iii)	=>	$I = acf$	(vi)

Equation 6-4 Generating and identity relations for Static-Agent and Mobile-Agent systems

$l(a) \rightarrow a + be + cf + (bcd)$	(i)	=>	$l(a) \rightarrow a + be + cf$	(vii)
$l(b) \rightarrow b + ae + (acd) + (abcf)$	(ii)	=>	$l(b) \rightarrow b + ae$	(viii)
$l(c) \rightarrow c + af + (abd) + (abce)$	(iii)	=>	$l(c) \rightarrow c + af$	(ix)
$l(d) \rightarrow d + (abc) + (abde) + (acdf)$	(iv)	=>	$l(d) \rightarrow d$	(x)
$l(e) \rightarrow e + ab + (acef) + (abcde)$	(v)	=>	$l(e) \rightarrow e + ab$	(xi)
$l(f) \rightarrow f + ac + (abef) + (abcdf)$	(vi)	=>	$l(f) => f + ac$	(xii)

Equation 6-5 Confounding patterns in initial fraction for Static-Agent and Mobile-Agent systems

The system is run with all combinations of the first 3 factors and the aliased values of the other factors. The results of these set of runs, also called the initial fraction, are given in Table 6-10 (Static-Agent) and Table 6-13 (Mobile-Agent). Since the main interactions of the factors cannot be gauged from these results due to the confounding effects of the higher order interactions, we redo the runs after reversing the sign of factor *a* i.e. changing +1 to -1 and -1 to +1. This process is called a fold-over operation and these set of runs are collectively called the fold-over fraction. This essentially means that we are conducting more experiments with new combinations of the factor values and using these new results to isolate the main effects. For the fold-over operations, we use a new set of dummy variables a', b', c', d', e' and f' as shown in Table 6-11 and Table 6-14. For the Static-Agent and Mobile-Agent systems, we flip the sign of

the dummy variable a and keep all the other dummy variable unchanged such that $a^{'} = -a,\ b^{'} = b,\ c^{'} = c,\ d^{'} = d, e^{'} = e$. Using these fold-over variables, new aliasing relationships between the main effects and the second-order effects are formed (Equation 6-6). By combining the aliasing relations for the initial and fold-over factorial designs (Equation 6-7), we are able to isolate the main interactions of the factors.

$$l(a') \rightarrow -a + be + cf \qquad \text{(i)}$$

$$l(b') \rightarrow b - ae \qquad \text{(ii)}$$

$$l(c') \rightarrow c - af \qquad \text{(iii)}$$

$$l(e') \rightarrow e - ab \qquad \text{(iv)}$$

$$l(f') \rightarrow f - ac \qquad \text{(v)}$$

Equation 6-6 Confounding patterns in fold-over fraction for Static-Agent and Mobile-Agent systems

$$\frac{1}{2}(l(a) - l(a^{'})) \rightarrow a \qquad \text{(i)}$$

$$\frac{1}{2}(l(b) + l(b^{'})) \rightarrow b \qquad \text{(ii)}$$

$$\frac{1}{2}(l(c) + l(c^{'})) \rightarrow c \qquad \text{(iii)}$$

$$\frac{1}{2}(l(e) + l(e^{'})) \rightarrow e \qquad \text{(iv)}$$

$$\frac{1}{2}(l(f) + l(f^{'})) \rightarrow f \qquad \text{(v)}$$

Equation 6-7 Combining initial and fold-over fractions in Static-Agent and Mobile-Agent systems

Factors		Dummy variables	Aliasing	Min Level (-1)	Max Level (+1)
Symbol	**Description**				
S_S	Similarity Distribution	a		0.2	0.8
M_S	No. of Agents	b		16	64
G_S	Grid Size (m x n)	c		2x2	4x4
T_S	Processing Time	d	d=abc	10s	50s
R_S	Entry Rate	e	e=ab	1 candidate/s	10 candidates/s
N_S	Search Space Size	f	f=ac	200	1000

Table 6-9 List of factors for Static-Agent system

$l(a)$	$l(b)$	$l(c)$	$l(d)$	$l(e)$	$l(f)$	Y	$l(a)*Y$	$l(b)*Y$	$l(c)*Y$	$l(d)*Y$	$l(e)*Y$	$l(f)*Y$
-1	-1	-1	-1	1	1	736.57	-736.57	-736.57	-736.57	-736.57	736.57	736.57
-1	-1	1	1	1	-1	2601.44	-2601.44	-2601.44	2601.44	2601.44	2601.44	-2601.44
-1	1	-1	1	-1	1	1676.87	-1676.87	1676.87	-1676.87	1676.87	-1676.87	1676.87
-1	1	1	-1	-1	-1	200	-200	200	200	-200	-200	-200
1	-1	-1	1	-1	-1	1111.55	1111.55	-1111.55	-1111.55	1111.55	-1111.55	-1111.55
1	-1	1	-1	-1	1	1292.04	1292.04	-1292.04	1292.04	-1292.04	-1292.04	1292.04
1	1	-1	-1	1	-1	88.85	88.85	88.85	-88.85	-88.85	88.85	-88.85
1	1	1	1	1	1	3387.28	3387.28	3387.28	3387.28	3387.28	3387.28	3387.28
						1386.83	166.21	-97.15	966.73	1614.92	633.42	772.73
							$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(f)$

Table 6-10 Fractional factorial analysis of Static-Agent system

$l(a')$	$l(b')$	$l(c')$	$l(d')$	$l(e')$	$l(f')$	Y	$l(a')*Y$	$l(b')*Y$	$l(c')*Y$	$l(d')*Y$	$l(e')*Y$	$l(f')*Y$
1	-1	-1	1	-1	-1	607.94	607.94	-607.94	-607.94	607.94	-607.94	-607.94
1	-1	1	-1	-1	1	2332.33	2332.33	-2332.33	2332.33	-2332.33	-2332.33	2332.33
1	1	-1	-1	1	-1	1472.64	1472.64	1472.64	-1472.64	-1472.64	1472.64	-1472.64
1	1	1	1	1	1	169.71	169.71	169.71	169.71	169.71	169.71	169.71
-1	-1	-1	-1	1	1	1273.42	-1273.42	-1273.42	-1273.42	-1273.42	1273.42	1273.42
-1	-1	1	1	1	-1	1514.29	-1514.29	-1514.29	1514.29	1514.29	1514.29	-1514.29
-1	1	-1	1	-1	1	93.65	-93.65	93.65	-93.65	93.65	-93.65	93.65
-1	1	1	-1	-1	-1	3662.54	-3662.54	3662.54	3662.54	-3662.54	-3662.54	-3662.54
						1390.82	-490.32	-82.36	1057.805	-1588.84	-566.6	-847.075
							$L(a')$	$L(b')$	$L(c')$	$L(d')$	$L(e')$	$L(f')$

Table 6-11 Fold-over operation for fractional factorial analysis of Static-Agent system

Factors		Dummy variables	Aliasing	Min Level (-1)	Max Level (+1)
Symbol	**Description**				
S_M	Similarity Distribution	a		0.2	0.8
M_M	No. of Agents	b		16	64
G_M	Grid Size (m x n)	c		2x2	4x4
T_M	Processing Time	d	d=abc	10s	50s
R_M	Entry Rate	e	e=ab	1 candidate/s	10 candidates/s
N_M	Search Space Size	f	f=ac	200	1000

Table 6-12 List of factors for Mobile-Agent system

$l(a)$	$l(b)$	$l(c)$	$l(d)$	$l(e)$	$l(f)$	Y	$l(a)*Y$	$l(b)*Y$	$l(c)*Y$	$l(d)*Y$	$l(e)*Y$	$l(f)*Y$
-1	-1	-1	-1	1	1	627.68	-627.68	-627.68	-627.68	-627.68	627.68	627.68
-1	-1	1	1	1	-1	784.52	-784.52	-784.52	784.52	784.52	784.52	-784.52
-1	1	-1	1	-1	1	1341.56	-1341.56	1341.56	-1341.56	1341.56	-1341.56	1341.56
-1	1	1	-1	-1	-1	194.73	-194.73	194.73	194.73	-194.73	-194.73	-194.73
1	-1	-1	1	-1	-1	790.43	790.43	-790.43	-790.43	790.43	-790.43	-790.43
1	-1	1	-1	-1	1	937.71	937.71	-937.71	937.71	-937.71	-937.71	937.71
1	1	-1	-1	1	-1	89.9	89.9	89.9	-89.9	-89.9	89.9	-89.9
1	1	1	1	1	1	884.53	884.53	884.53	884.53	884.53	884.53	884.53
						706.38	-61.48	-157.405	-12.02	487.76	-219.45	482.97
							$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(f)$

Table 6-13 Fractional factorial analysis of Mobile-Agent system

$l(a')$	$l(b')$	$l(c')$	$l(d')$	$l(e')$	$l(f')$	Y	$l(a')*Y$	$l(b')*Y$	$l(c')*Y$	$l(d')*Y$	$l(e')*Y$	$l(f')*Y$
1	-1	-1	1	-1	-1	393.51	393.51	-393.51	-393.51	393.51	-393.51	-393.51
1	-1	1	-1	-1	1	865.26	865.26	-865.26	865.26	-865.26	-865.26	865.26
1	1	-1	-1	1	-1	969.82	969.82	969.82	-969.82	-969.82	969.82	-969.82
1	1	1	1	1	1	169.46	169.46	169.46	169.46	169.46	169.46	169.46
-1	-1	-1	-1	1	1	875.74	-875.74	-875.74	-875.74	-875.74	875.74	875.74
-1	-1	1	1	1	-1	983.31	-983.31	-983.31	983.31	983.31	983.31	-983.31
-1	1	-1	1	-1	1	86.23	-86.23	86.23	-86.23	86.23	-86.23	86.23
-1	1	1	-1	-1	-1	868.62	-868.62	868.62	868.62	-868.62	-868.62	-868.62
						651.49	651.4938	-103.96	-255.92	140.33	-486.73	196.17
							$L(a')$	$L(b')$	$L(c')$	$L(d')$	$L(e')$	$L(f')$

Table 6-14 Fold-over operation for fractional factorial analysis of Mobile-Agent system

Similarly, we use a 2_{IV}^{5-2} factorial design for the Baseline system. The factors used and its dummy variables are depicted in Table 6-15. This design uses two generator and identity relations shown in Equation 6-8. From analyzing the confounding patterns in Equation 6-9, we see that only the main effects of c and d can be calculated directly using the runs in the initial fraction. The effects of the remaining factors are confounded by higher-order interactions. A fold-over operation is conducted by flipping the sign of the variable a. The confounding patterns for the fold-over fraction are given in Equation 6-10. The results of the initial fraction and fold-over fraction runs are used with the relations in Equation 6-11, to calculate the main effects.

$d = abc$	(i)	=>	$I = abcd$	(iii)
$e = ab$	(ii)	=>	$I = abe$	(iv)

Equation 6-8 Generator and identity relations for Baseline system

$l(a) \rightarrow a + be + (bcd)$	(i)	=>	$l(a) \rightarrow a + be$	(vi)
$l(b) \rightarrow b + ae + (acd)$	(ii)	=>	$l(b) \rightarrow b + ae$	(vii)
$l(c) \rightarrow c + (abd) + (abce)$	(iii)	=>	$l(c) \rightarrow c$	(viii)
$l(d) \rightarrow d + (abc) + (abed)$	(iv)	=>	$l(d) \rightarrow d$	(ix)
$l(e) \rightarrow e + ab + (abcde)$	(v)	=>	$l(e) \rightarrow e + ab$	(x)

Equation 6-9 Confounding patterns in initial fraction for Baseline system

$l(a') \rightarrow -a + be$	(i)
$l(b') \rightarrow b - ae$	(ii)
$l(e') \rightarrow e - ab$	(iii)

Equation 6-10 Confounding patterns in fold-over fraction for Baseline system

$$\frac{1}{2}\left(l(a) - l(a')\right) \rightarrow a \qquad \text{(i)}$$

$$\frac{1}{2}\left(l(b) + l(b')\right) \rightarrow b \qquad \text{(ii)}$$

$$\frac{1}{2}\left(l(e) + l(e')\right) \rightarrow e \qquad \text{(iii)}$$

Equation 6-11 Combining initial and fold-over fractions in Baseline system

Factors		**Dummy variables**	**Aliasing**	**Min Level (-1)**	**Max Level (+1)**
Symbol	**Description**				
M_B	No. of Agents	a		16	64
R_B	Entry Rate	b		1 candidate/s	10 candidates/s
N_B	Search Space Size	c		200	1000
T_B	Processing Time	d	d = abc	10s	50s
S_B	Similarity Distribution	e	e = ab	0.2	0.8

Table 6-15 List of factors for Baseline system

$\boldsymbol{m(a)}$	$\boldsymbol{m(b)}$	$\boldsymbol{m(c)}$	$\boldsymbol{m(d)}$	$\boldsymbol{m(e)}$	$\boldsymbol{Y}$	$\boldsymbol{m(a) * Y}$	$\boldsymbol{m(b) * Y}$	$\boldsymbol{m(c) * Y}$	$\boldsymbol{m(d) * Y}$	$\boldsymbol{m(e) * Y}$
-1	-1	-1	-1	1	196.26	-196.26	-196.26	-196.26	-196.26	196.26
-1	-1	1	1	1	4273.16	-4273.16	-4273.16	4273.16	4273.16	4273.16
-1	1	-1	1	-1	648.31	-648.31	648.31	-648.31	648.31	-648.31
-1	1	1	-1	-1	3030.46	-3030.46	3030.46	3030.46	-3030.46	-3030.46
1	-1	-1	1	-1	315.81	315.81	-315.81	-315.81	315.81	-315.81
1	-1	1	-1	-1	3307.87	3307.87	-3307.87	3307.87	-3307.87	-3307.87
1	1	-1	-1	1	206.21	206.21	206.21	-206.21	-206.21	206.21
1	1	1	1	1	3908.82	3908.82	3908.82	3908.82	3908.82	3908.82
					1985.863	-102.37	-74.825	3288.43	601.325	320.5
						$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$

Table 6-16 Fractional factorial analysis of Baseline system

$m(a')$	$m(b')$	$m(c')$	$m(d')$	$m(e')$	Y	$m(a') * Y$	$m(b') * Y$	$m(c') * Y$	$m(d') * Y$	$m(e') * Y$
-1	-1	-1	-1	1	196.26	-196.26	-196.26	-196.26	-196.26	196.26
-1	-1	1	1	1	4273.16	-4273.16	-4273.16	4273.16	4273.16	4273.16
-1	1	-1	1	-1	648.31	-648.31	648.31	-648.31	648.31	-648.31
-1	1	1	-1	-1	3030.46	-3030.46	3030.46	3030.46	-3030.46	-3030.46
1	-1	-1	1	-1	315.81	315.81	-315.81	-315.81	315.81	-315.81
1	-1	1	-1	-1	3307.87	3307.87	-3307.87	3307.87	-3307.87	-3307.87
1	1	-1	-1	1	206.21	206.21	206.21	-206.21	-206.21	206.21
1	1	1	1	1	3908.82	3908.82	3908.82	3908.82	3908.82	3908.82
					1985.86	-102.37	-74.825	3288.43	601.325	320.5
						$L(a')$	$L(b')$	$L(c')$	$L(d')$	$L(e')$

Table 6-17 Fold-over operation for fractional factorial analysis of Baseline system

6.4.4 Results

The main effects of the factors calculated from the factorial analysis are listed in Table 6-18. We analyze these values and venture to provide explanations for the trends seen in the experimental results.

1. **No. of Agents (M_F)**: We see that this factor has a beneficial effect on the total-recall time in all the three systems. This means that an increase in the number of agents leads to a decrease in the total-recall time. This effect is easy to understand – more resource, less time for searching. The greatest rate of decrease is seen in the Mobile-Agent system, followed by the Static-Agent system and then by the Baseline system. Due to the dynamic nature of its resource allocation, the Mobile-Agent system is able to efficiently deploy the resources by avoiding the negative effects of agent idleness and agent contention. Agent contention is a major concern in the Baseline system as all the agents try to process the candidate with the highest partial-values. Although the increase in the number of agents leads to better performance in static agent system, its performance is stunted by the problem of agent idleness.
2. **Grid-Size (G_F)**: The performance of the Static-Agent system decreases sharply with increase in the number of grids. One reason for this is that with the increase in the grid-size, the average number of candidates per agent decreases leading to more agent idleness. This increased sparsity of candidates has a smaller negative effect in the Mobile-Agent system due to its ability to move the agents around.

Despite the dynamic allocation of resources in the Mobile-Agent, there is a small increase in the recall time with grid-size. This is due to restriction imposed by the Mobile-Agent system implementation – agents can move only to its immediate neighborhood. Under increasing sparsity of the grids, the chances that migrating agents will land in an empty grid-block increases, leading to agents being idle. Since Baseline does not use grids, this factor does not affect its performance.

3. **Processing Time (T_F)**: is the time it takes to fully process a candidate. It is clear that larger candidates should increase the recall time. How does the characteristic of each system help in mitigating this problem is the question? Among the control systems, the Mobile-Agent system performs best with increase in T_F. We also see that the Static-Agent system performs the worst. Due to the immovability of the agents in the Static-Agent system, there is larger likelihood for some grids to be lightly loaded and others to be heavily loaded. Since each grid can have only fixed amount of resource, having larger candidates exacerbates the problem. Surprisingly, Baseline does not do too bad compared to the Mobile-Agent system. The reason is that with small documents, the agents do the candidate selection procedure at a faster rate leading to greater agent contention. So as the size of the candidates increase, the contention decreases. This has a positive effect on the performance of the Baseline system.

Factors		**Baseline**		**Static-Agent**		**Mobile-Agent**	
Symbol	**Description**	Mean	Stdev	Mean	Stdev	Mean	Stdev
S_F	Similarity Dist.	3.75	24.48	314.52	13.19	27.36	6.51
M_F	No. of Agents	-22.94	7.12	-83.91	5.11	-192.77	15.22
G_F	Grid Size	NA	NA	1019.23	9.57	73.52	21.17
T_F	Processing Time	558.94	42.09	1604.98	8.63	503.85	36.33
R_F	Entry Rate	-43.14	25.72	33.04	2.24	-14.53	20.18
N_F	Search Space Size	3332.99	50.06	-34.88	5.67	76.22	18.89

Table 6-18 Comparison of Main Effects for the Control Systems

4. **Entry Rate (R_F)**: It may be counter-intuitive to note that the performance does not vary a lot with the entry rate. This is partly due to the design of the simulation. Since the simulation is conducted with a relatively small amount of candidates [200-1000], the true effect of the entry rate is not seen. When

the rate is high, the candidates get into the search space quickly and the search space is relatively static after that. Hence its effect is comparatively less when compared with other factors.

5. **Similarity Distribution (S_F)**: A higher value of S_F means that there are larger numbers of candidates with high final values. This means that there is a higher chance of the system allocating resources to candidates that do not turn out to be a relevant result. This factor affects the Static-Agent system and the Mobile-Agent system more than the Baseline system. The reason is that with high value of S_F a greater proportion of the candidates tend to congregate at certain grid-blocks (in the large partial-value regions). Hence, there is a load imbalance in the grid space and such load imbalances affect the Static-Agent system more than the Mobile-Agent system. Since the Mobile-Agent system uses local load information to move the agents, it is able to mitigate the effects of this factor. The Baseline does not use a grid-system and so its performance is not affected.
6. **Search Space Size (N_F)**: In the Baseline system, we see a large decrease in performance with the increase in the search size. Since the candidates initially have a priority value of zero in this system, the Baseline process has a better chance of processing candidates fully before moving to another. Hence when the search size increases, the time it takes to get to the relevant candidates is higher and this is the reason why its performance decreases with search space size. Having a grid space which categorizes the candidates using its partial-values helps the agents to get to the top candidates quicker. Therefore the effect of this parameter is much less on the Static-Agent and Mobile-Agent systems.

6.4.5 Summary

The characteristics of candidates in the search space and the parameters of the search space model have an effect on the efficiency of the algorithms implemented within the various components in the framework. In experiment-2, we used detailed statistical study to analyze the performance of the Mobile-Agent system. By using multiple testbeds, we have shown that resource allocation based on our search space model performs better than the traditional resource allocation methodologies in a clear majority of

the cases. We have also demonstrated the ability of the resource allocation framework to support detailed performance analysis.

6.5 Experiment-3: Resource Allocation with Network Dynamism Models

Although experiments 1 & 2 validated the search space model, with its notions of grouping candidates into explorative and exploitative regions, explicit modeling of dynamism was left out. Using the ideas of network flows, we can model both the internal and external dynamism in the search space, and approximate the future conditions in the network. In this experiment we validate this methodology by showing that resource allocation strategies that use this dynamism model will lead to better performance.

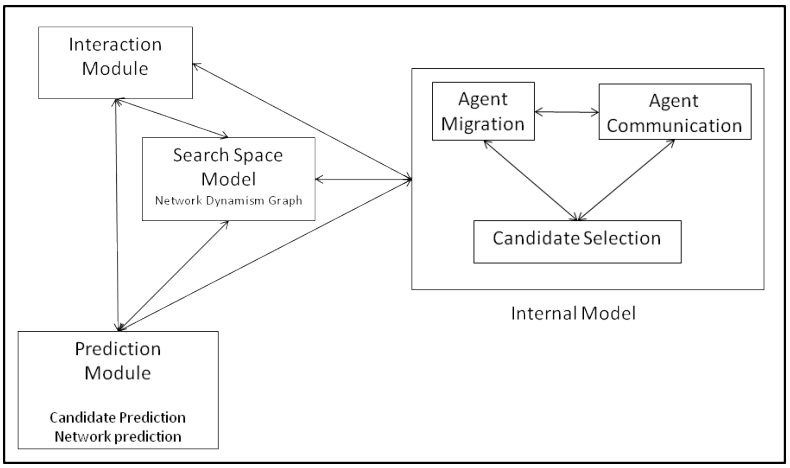

Figure 6-9 Dynamic-Agent system

6.5.1 Control Systems

For validation, we compare the Mobile-Agent system from experiments 1&2 with a control system called the Dynamic-Agent system (Figure 6-9). The Dynamic-Agent system differs from the Mobile-Agent system only in the way it decides on agent migration. Mobile-Agent system uses only the current load conditions in the neighborhood to decide on the grid-block to move to. On the other hand, Dynamic-Agent system implements the network dynamism graph and updates the flow rates of candidates and agents. Using the flow rates with the load prediction algorithm (described in Section 6.1.4), the future

values of the load metrics are calculated. In the Dynamic-Agent system, the predictions of future load conditions are used as an input to the agent migration sub-component. This information is used along with the result quality and current load information to make decisions on whether to move to a new grid-block or not.

6.5.2 Simulation Setup

We compare the performance of the Dynamic-Agent system with the Mobile-Agent system that was used in experiments 1 & 2. We run these two control machines on a large testbed of 2000 candidates. The testbeds are also made more dynamic by introducing groups of 100 candidates at regular intervals of time. The Dynamic-Agent system uses a history of the flow rate values of candidates and agents to predict future values. Hence its performance is expected to improve only gradually and this trend is what we hope to see in the experimental results. This is the reason for using a larger testbed and simulating it over a longer period of time. We introduce a set of 10 relevant candidates at regular intervals in the search space. For each of these sets of relevant candidates, we calculate the proportion of these candidates that are discovered first by the control systems. We call this performance metric the Discovery-Rate of the system. We run the Mobile-Agent and the Dynamic-Agent system on the testbeds. For each relevant set of 10 candidates, we note the proportion discovered first by each system. Both the Mobile-Agent and Dynamic-Agent systems are made to run on each testbed 3 times and the Discovery-Rate values are tabulated. We use three testbeds which differ in the proportion of small and large candidates contained in it. This is represented by the parameter β which is defined as the ratio of small documents to large documents. Large candidates have a processing time of 50s and small candidates have a processing time of 10s. Since the small candidates get processed quickly, the internal dynamism within the search space will be high when the proportion of small candidates in the search space is large. By using testbeds with different β values, we will be able to test the control systems under different rates of internal dynamism.

6.5.3 Results

The discovery-rate of the Dynamic-Agent system are plotted against the set of relevant candidates for β values 0.1, 0.2 and 0.3 in Figure 6-10, Figure 6-11 and Figure 6-12 respectively. In these graphs, the relevant sets of candidates are labeled 1, 2 and so on. Each set contains 10 candidates. From the graph plots, we can see that the discovery rate of the system is generally low in the beginning of the simulation and gradually improves over the simulation period. This is because in the initial stages, the prediction module does not have enough samples of the network condition, in order to make good predictions. We also see that the upward trend of the discovery-rate is clearer with low β values. In other words, the load prediction algorithms work well with low internal dynamism rates. The reason for this is that the horizon of prediction in the load prediction algorithm remains constant. Such coarse grained approximation will perform worse as the flow rates fluctuate rapidly with increases in internal dynamism. With these results, we have demonstrated the capability of our dynamism models to support prediction of network conditions using the concepts of internal and external dynamism. Developing more sophisticated prediction algorithms based on the dynamism models is part of the future work.

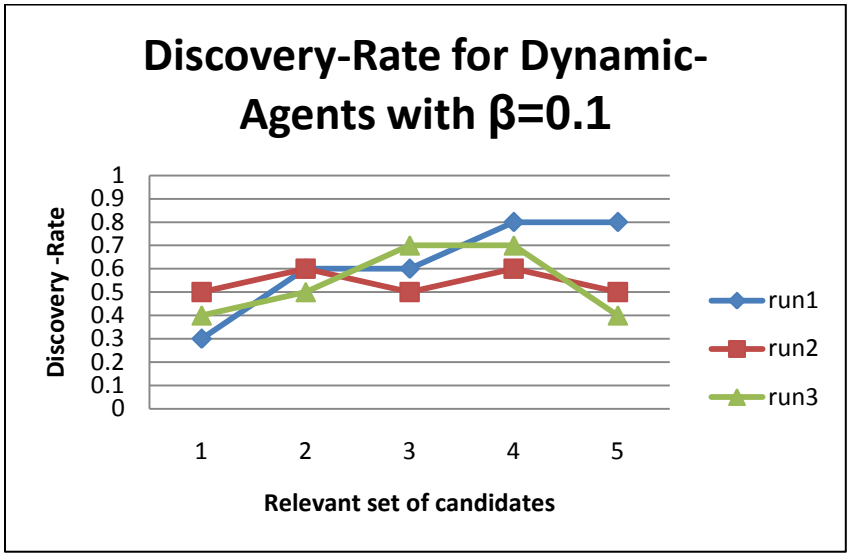

Figure 6-10 Discovery-rate for Dynamic-Agent system with β = 0.1

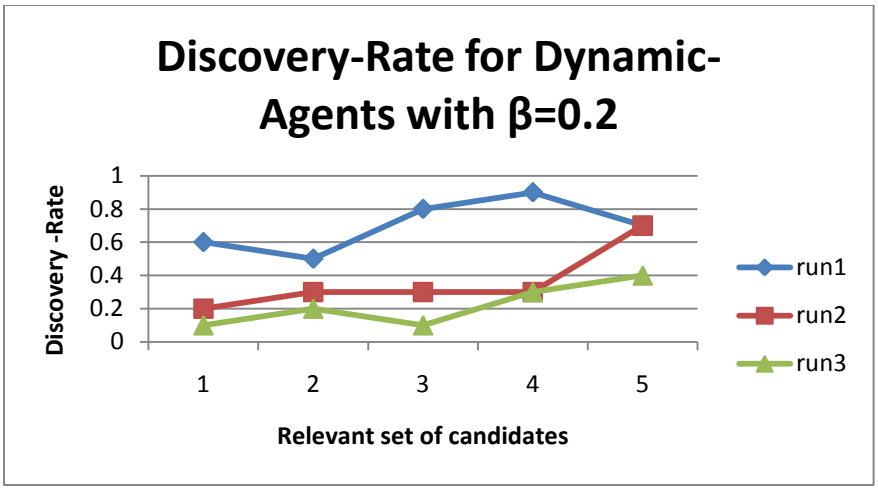

Figure 6-11 Discovery-rate for Dynamic-Agent system with β = 0.2

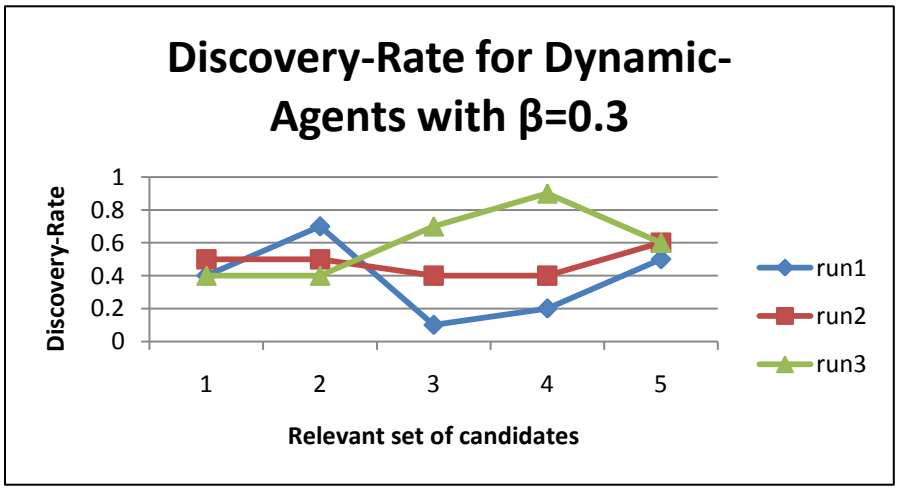

Figure 6-12 Discovery-rate for Dynamic-Agent system with β = 0.3

6.6 Summary

We validated two essential aspects of the framework that was presented in the previous chapter – grid based search space model and the notion of internal dynamism. Due to the large number of factors and the wide range of their values, we adopted a statistical study based on factorial design to determine their effects on the performance of the Mobile-Agent and other control systems. We also formulated flow metrics to capture the internal and external dynamism in the search space and validated their effectiveness

in providing better resource allocation. Essentially, we have demonstrated the versatility of our framework by both designing resource allocation strategies and analyzing its performance.

Chapter 7

Conclusion

Information retrieval in modern search spaces like the internet has become a challenge due to its size and the growing prevalence of real time information. Current research in information retrieval has focused heavily on improving the precision and recall performance of retrieval algorithms and not so much on timely retrieval of dynamic information. In this dissertation, we proposed a novel approach towards real time retrieval by making intelligent, multi-grained resource allocation the center piece of the solution. The relevancy approximations provided by partial processing is key to the success of using such resource allocation. In the concluding chapter, we list the main contributions made in the dissertation followed by possible future extensions to this work.

7.1 Main Contributions

The work presented in this dissertation has made contributions in the fields of Content Based Image Retrieval (CBIR) and real time multiagent resource allocation. By developing an anytime image retrieval algorithm, we have presented a real time image retrieval methodology that uses intelligent resource allocation for quick retrieval of relevant images. Partial processing has been shown to be viable for online processing of large and dynamic document spaces. However, the rapid changes in the search space brought on about by varying partial-values impart an extra layer of complexity to the problem. In order to design effective resource allocation strategies, we model both external and internal dynamism of the search space within a generic multiagent based resource allocation framework.

1. **Image Retrieval**: The partial processing based image retrieval algorithm presented in this dissertation maps low level features to high level concepts. This is in keeping with the current research direction in image retrieval towards identifying human-understandable concepts using mathematical representation of the image. Our main contribution has been in: 1) using text query

instead of image query, 2) learning techniques in concept matching, and 3) concept library. Using text queries with image databases is a difficult problem as this requires converting information from one form to the other. Our CBIR based algorithm converts the information from the text query and images into concept graphs which act as a common knowledge representation. Use of concept graphs had been proposed to tackle data heterogeneity in the work on I-FGM and was validated with text documents. In this dissertation, we have been able to extend its use to complicated data types such as images. Although image algorithms that match low level features with concepts have been proposed earlier, our contribution has been in the way we use training data to learn the common features of a particular concept. Our multi-instance learning technique for forming the concept signature was validated in this dissertation. By using a text description for each concept, we also incorporate contextual information in the image concepts. This is in contrast to current techniques that use single word concepts.

2. **Real Time Image Retrieval**: Partial processing is central to intelligent resource allocation strategies described in this dissertation. Although partial processing based text retrieval algorithm had been validated in the work on I-FGM, its utility with other complicated data types was still in question. We presented an anytime, CBIR based image retrieval algorithm that is able to calculate intermediate partial similarity values while converging to the final similarity value. Interpreting partial visual information is a difficult problem that we tackled using this algorithm. By validating its performance with the state of the art and by deploying intelligent resource allocation with the image algorithm, we demonstrate a wider utility for partial processing. By doing so, we have also made contributions to the application of anytime algorithms in information retrieval.

3. **Resource Allocation Framework:** Efficient real time resource allocation is important for providing the kind of timely retrieval that is expected with large and dynamic databases. Current multiagent frameworks for real time retrieval have tended to incorporate classic methods such as game theory [**194**] and Markov methods [**42**] without modeling the underlying dynamism. We propose a multiagent resource allocation framework which models the important characteristics

of the complex and dynamic search space such as dynamism while remaining generic enough to be able to support various resource allocation methodologies. The important aspects of the framework are:

a. **Search Space Model**: The search space model that represents the various parameters of the documents is the lynchpin of our framework. One of the main contributions of the model is its ability to provide a coarse grained state representation of the search space by using a grid-based partitioning. Although grid-based search space models have been used before, we have been able to model real time changes in the search space by representing the flow of agents and candidates between the grid-blocks. This provides a concise and realistic representation of the dynamism in the search space.

b. **Modeling Dynamism**: Since resource allocation algorithms for real time search should quickly re-prioritize their allocation based on changes in the search space caused by entry of new documents and exit of old ones, modeling the dynamism in the search space is critical. Modeling only the external dynamism is insufficient when partial processing is involved as the partial similarity of the document is also continuously changing. The most important contribution of this dissertation is introducing a novel search space model that also represents the changes in the partial similarity or internal dynamism. By using a grid-based partitioning of the search space and using flow based metrics to quantify internal and external dynamism, we are able to produce a coarse grained representation of an otherwise complex environment and mitigate the effects of partial observability.

c. **Flexible Architecture:** The framework is component-based to encapsulate various aspects of agent internal model and agent interactions in multiple components. Various resource allocation methods can be plugged into the framework with minimal system wide changes. The dissertation also shows how various combinations of the components can be used to represent important behavior of the search space.

7.2 Future Work

We have listed some of the interesting avenues for future extensions of the work in this dissertation.

1. **Improving Concept Matching**: With large concept graphs, matching the image concepts with each concept in the library can take long. Solving this problem is important for preserving the real time nature of the retrieval process. One possible solution is to use a hierarchical concept library where more generic concepts such as "automobile" are higher up in the hierarchy than specific concepts such as "Toyota". With such hierarchical concepts, image concepts can be first matched with generic concepts to filter out the obvious mismatches. The image can then be compared with more specific concepts in these initial matches to get more accurate concepts matches.
2. **Dynamically Changing Documents**: Another future avenue is to look at dynamic documents – documents that get updated with new information over a period of time. Reprocessing the documents from scratch with each update is not efficient. Anytime-anywhere algorithms [**195**] have shown great promise in efficiently dealing with such type of dynamism. An anytime algorithm is said to have the anywhere property when it can utilize partial results generated elsewhere (even a different algorithm) in order to create better solutions. Anytime-anywhere algorithms have been used to deal with dynamism in social network analysis [**195**] [**196**]. Developing such algorithms within the partial processing paradigm to deal with these issues is useful for multimedia documents such as video where the documents are continuously updated with new streaming data.
3. **Resource Allocation Framework**: In the chapters on the resource allocation framework, the experimental results validated only a limited aspect of the framework. The framework can also be used to design and analyze more complex agent behavior. For example, agents can utilize better prediction algorithms that model the stochastic nature of the search space. The grid search space model can also be extended to incorporate dynamically changing grids, which in turn affects the granularity of resource allocation. When the number of grid-blocks is small, the

resource allocation is coarse grained and vice-versa. The grids can also be non-homogeneous with some regions in the search space being more finely divided than others. An analysis of how the grids change with dynamism metrics can lead to resource allocation strategies that can react more effectively to changes in the search space.

Bibliography

[1] T. J. Berners-Lee, R. Cailliau, and J. F. Groff, "The world-wide web," *Computer networks and ISDN systems*, vol. 25, no. 4-5, p. 270, 1992.

[2] O. Frieder, D. Grossman, and A. Chowdhu, "Efficiency Considerations in Very Large Information Retrieval Servers," *Journal of Digital Information (British Computer Society)*, vol. 1, pp. 1-4, 1999.

[3] A. Singhal, "Modern Information Retrieval: A Brief Overview," *IEEE Data Engineering Bulletin*, vol. 24, no. 4, pp. 35-43, 2001.

[4] R. Baeza-Yates, "Information retrieval in the Web: beyond current search engines," *International Journal of Approximate Reasoning*, vol. 34, no. 2-3, pp. 97-104, Nov. 2003.

[5] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[6] M. Gordon and P. Pathak, "Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines," *Information processing & management*, vol. 35, no. 2, p. 141, 1999.

[7] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of the 7th International WWW Conference*, 1998, pp. 107-117.

[8] M. Kirkpatrick. (2009, May) ReadWriteWeb.[Online]. http://www.readwriteweb.com/archives/larry_page_on_real_time_google_we_have_to_do_it.php

[9] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international Conference on World Wide Web*, Banff, Alberta, Canada, 2007, pp. 271-280.

[10] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, p. 6, 2006.

[11] R. Lempel, et al., "Just in time indexing for up to the second search," in *Proceedings of the Sixteenth ACM Conference on Conference on information and Knowledge Management*, Lisbon, Portugal, 2007, pp. 97-106.

[12] V. Ercegovac, V. Josifovski, N. Li, M. R. Mediano, and E. J. Shekita, "Supporting sub-document updates and queries in an inverted index," in *Proceeding of the 17th ACM Conference on information and Knowledge Management*, Napa Valley, California, 2008, pp. 659-668.

[13] M. Boddy and T. Dean, "Solving Time-Dependent Planning Problems," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Menlo Park, CA, 1989, p. 979–984.

[14] E. Santos Jr., S. E. Shimony, and E. M. Williams, "Solving Hard Computational Problems through Collections (Portfolios) of Cooperative Heterogeneous Algorithms," in *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, 1999, pp. 356-360.

[15] R. Dolin, D. Agrawal, A. El Abbadi, and L. Dillon, "Pharos: a scalable distributed architecture for locating heterogeneous information sources," in *Proceedings of the Sixth international Conference on information and Knowledge Management CIKM '97*, New York, 1997, pp. 348-355.

[16] L. Shklar, A. Sheth, V. Kashyap, and K. Shah, "InfoHarness: Use of Automatically Generated

Metadata for Search and Retrieval of Heterogeneous Information," *Lecture Notes in Computer Science*, pp. 217-230, 1995.

[17] R. K. Srihari, Z. Zhang, and A. Rao, "Intelligent Indexing and Semantic Retrieval of Multimodal Documents," *Information Retrieval*, vol. 2, no. 2, pp. 245-275, May 2000.

[18] M. T. Martin-Valdivia, M. C. Diaz-Galiano, A. Montejo-Raez, and L. A. Urena-Lopez, "Using information gain to improve multi-modal information retrieval systems," *Information Processing & Management*, vol. 44, no. 3, pp. 1146-1158, May 2008.

[19] E. Santos Jr, H. Nguyen, and M. S. Brown, "Kavanah: an Active User Interface Information Retrieval Agent Technology," in *Proceedings of the 2nd Asia-Pacific Conference on Intelligent Agent Technology*, Maebashi, Japan, 2001, pp. 412-423.

[20] E. Santos Jr., H. Nguyen, Q. Zhao, and E. Pukinskis, "Empirical Evaluation of Adaptive User Modeling in a Medical Information Retrieval Application," in *Lecture Notes in Computer Science Vol. 2702, User Modeling 2003*. Berlin / Heidelberg: Springer , 2003, p. 148.

[21] M. L. Littman, "A tutorial on partially observable Markov decision processes," *Journal of Mathematical Psychology*, vol. 53, pp. 119-125, 2009.

[22] U. Hanani, B. Shapira, and P. Shoval, "Information Filtering: Overview of Issues, Research and Systems," *User Modeling and User-Adapted Interaction*, vol. II, no. 3, pp. 203-259, 2001.

[23] A. Oerlemans, J. T. Rijsdam, and M. S. Lew, "Real-time object tracking with relevance feedback," in *Proceedings of the 6th ACM international Conference on Image and Video Retrieval*, Amsterdam, The Netherlands, 2007, pp. 101-104.

[24] E. Santos Jr., E. Santos, H. Nguyen, L. Pan, and J. Korah, "Large-scale Distributed Foraging, Gathering, and Matching for Information Retrieval: Assisting the Geospatial Intelligent Analyst," in *Proceedings of SPIE*, vol. 5803, 2005, pp. 66-77.

[25] E. Santos Jr., E. E. Santos, H. Nguyen, L. Pan, J. Korah, Q. Zhao, and M. Pittkin, "I-FGM Information Retrieval in Highly Dynamic Search Spaces," in *Proceedings of SPIE*, vol. 6229, 2006, p. 622901.

[26] E. Santos Jr., E. Santos Jr., E. E. Santos, H. Nguyen, L. Pan, J. Korah, Q. Zhao, and H. Xia, "Applying I-FGM to Image Retrieval and an I-FGM System Performance Analyses," in *Proceedings of the SPIE*, vol. 6560, Orlando, FL, 2007, p. 65600I.

[27] E. Santos Jr., E. E. Santos, H. Nguyen, L. Pan, J. Korah, H. Xia, F. Yu, and D. Li, "Analyst-Ready Large Scale Real Time Information Retrieval Tool for E-Governance," in *E-Government Diffusion, Policy and Impact: Advanced Issues and Practices*, M. Khosrow-Pour, Ed. IGI Global, 2008, pp. 268-294.

[28] E. Santos Jr., E. E. Santos, H. Nguyen, L. Pan, J. Korah, and H. Xia, "I-FGM as a Real Time Information Retrieval Tool for E-Governance," *International Journal of Electronic Government Research*, vol. 4, no. 1, pp. 14-35, 2008.

[29] P. Enser and C. Sandom, "Towards a comprehensive survey of the semantic gap in visual image retrieval," *Lecture Notes in Computer Science*, vol. 2728, pp. 291-299, 2003.

[30] S. Büttcher, C. L. Clarke, and B. Lushman, "Hybrid index maintenance for growing text collections," in *Proceedings of the 29th Annual international ACM SIGIR Conference on Research and Development in information Retrieval*, Seattle, WA, 2006, pp. 356-363.

[31] N. Lester, J. Zobel, and H. Williams, "Efficient online index maintenance for contiguous inverted lists," *Information Processing and Management*, vol. 42, no. 4, p. 916–933, Jul. 2006.

[32] R. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, and F. Silvestri, "Challenges in distributed information retrieval," in *Proceedings of the 23rd International Conference on Data Engineering*, 2007, pp. 6-20.

[33] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.

[34] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*, First Edition ed. Cambridge University Press, 2008.

[35] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. New York: Addison Wesley, 1999.

[36] N. Fuhr, "Probabilistic models in information retrieval," *The Computer Journal*, pp. 243-255, 1992.

[37] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st Annual international ACM SIGIR Conference on Research and Development in information Retrieval* , Melbourne, Australia, 1998, pp. 275-281.

[38] W. Meng, C. Yu, and K. Liu, "Building efficient and effective metasearch engines," *ACM Computing Surveys*, vol. 34, no. 1, pp. 48-89, Mar. 2002.

[39] J. Callan, "Distributed information retrieval," in *Advances in Information Retrieval*, W. B. Croft, Ed. Kluwer Academic Publishers, 2000, pp. 127-150.

[40] D. A. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, 2nd ed.

Springer, 2004.

[41] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, USA: SIAM, 1992.

[42] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proceedings of the 6th Conference on theoretical Aspects of Rationality and Knowledge*, San Francisco, 1996, pp. 195-210.

[43] S. Koenig, "Agent-Centered Search," *Artificial Intelligence Magazine*, vol. 22, no. 4, pp. 109-131, 2001.

[44] E. Santos Jr, E. E. Santos, L. Pan, and J. Korah, "A Large-Scale Distributed Framework for Information Retrieval in Large Dynamic Search Spaces," *Journal of Applied Intelligence*, To Appear.

[45] J. Korah, E. E. Santos, and E. Santos Jr, "A Multiagent Resource Allocation Framework with Partial Processing," Department of Computer Science, Virginia Tech, Blacksburg, Technical Report LCID-2009-201, 2009.

[46] D. L. Lee, Y. M. Kim, and G. Patel, "Efficient signature file methods for text retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 3, p. 423–435, Jun. 1995.

[47] R. Bayer and E. M. McCreight, "Organization and maintenance of large ordered indexes," *Acta Informatica*, vol. 1, no. 3, pp. 173-189, 1972.

[48] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted files versus signature files for text indexing," *ACM Transactions on Database Systems*, vol. 23, no. 4, p. 453–490, Dec. 1998.

[49] J. Cho and H. Garcıa-Molina, "The evolution of the web and implications for an incremental

crawler," in *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB2000)*, 2000, p. 200–209.

[50] S. Melnik, S. Raghavan, B. Yang, and H. Garcia-Molina, "Building a distributed full-text index for the web," in *Proceedings of the 10th International WWW*, 2001, p. 396–406.

[51] C. Badue, R. Baeza-Yates, B. Ribeiro-Neto, and N. Ziviani, "Distributed query processing using partitioned inverted files," in *Proceedings of the 9th String Processing and Information Retrieval Symposium (SPIRE)*, 2002.

[52] B. A. Ribeiro-Neto, E. S. d. Moura, M. S. Neubert, and N. Ziviani, "Efficient distributed algorithms to build inverted files," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR'99*, Berkeley, CA, 1999, p. 105–112.

[53] S. Melnik, S. Raghavan, and B. Yang, "Building a distributed full-text index for the web," in *Proceedings of WWW*, 2001, p. 396–406.

[54] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proceedings of OSDI '04: 6th Symposium on Operating System Design and Implemention*, San Francisco, CA, 2004.

[55] B. He, M. Patel, Z. Zhang, and K. C. Chang, "Accessing the deep web," *Communications of the ACM*, vol. 50, no. 5, pp. 94-101, May 2007.

[56] E. Selberg and O. Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler," in *Proceedings of the 4th International World Wide Web Conference*, 1995, pp. 195--208.

[57] J. A. Aslam and M. Montague, "Models for metasearch," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01*, New Orleans, Louisiana, 2001, p. 276–284.

[58] N. Craswell, "Methods for distributed information retrieval," PhD thesis, Department of Computer Science, The Australian Nation University, 2000.

[59] L. Si and J. Callan, "Modeling search engine effectiveness for federated search," in *Proceedings of the 28th Annual international ACM SIGIR Conference on Research and Development in information Retrieval*, Salvador, Brazil, 2005, pp. 83-90.

[60] L. Gravano and H. García-Molina, "Generalizing GLOSS to vector-space databases and broker," in *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, 1995.

[61] H. Nottelmann and N. Fuhr, "Evaluating different methods of estimating retrieval quality for resource," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[62] P. Ipeirotis and L. Gravano, "When one sample is not enough: improving text database selection using shrinkage," in *Proceedings of the ACM-SIGMOD International Conference on Management of Data* , 2004.

[63] F. Cacheda, V. Carneiro, and V. Plachoura, "Performance analysis of distributed information retrieval architectures using an improved network simulation model, Information Processing & Management," *Information Processing & Management*, vol. 43, no. 1, pp. 204-224, Jan. 2007.

[64] S. Russell and P. Norvig, *Artificial Intelligence: Modern Approach*, 2nd ed. Prentice Hall, 2002.

[65] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72-93, 2005.

[66] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts," *Multimedia Systems*, vol. 9, no. 2, pp. 170-184, Aug. 2003.

[67] T. Chothia and K. Chatzikokolakis, "A Survey of Anonymous Peer-to-Peer File-Sharing," in *Lecture Notes in Computer Science, Embedded and Ubiquitous Computing*. Berlin / Heidelberg: Springer, 2005, pp. 744-755.

[68] S. Androutsellis-Theotokis, "A survey of peer-to-peer file sharing technologies," Athens University of Economics and Business White Paper, 2002.

[69] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 161-172, 2001.

[70] I. Stoica, et al., "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *EEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17-32, Feb. 2003.

[71] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Lecture Notes in Computer Science, Designing Privacy Enhancing Technologies*. Berlin / Heidelberg: Springer, 2001, pp. 46-66.

[72] K. Aberer, F. Klemm, M. Rajman, and J. Wu, "An Architecture for Peer-to-Peer Information Retrieval," in *Proceedings of the 7th Annual International ACM SIGIR Conference, Workshop on Peer-to-Peer Information Retrieval*, 2004.

[73] J. Lu and J. Callan, "Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks," in *Lecture Notes in Computer Science, Advances in Information Retrieval*. Berlin / Heidelberg: Springer, 2005, pp. 52-66.

[74] H. Nottelmann and N. Fuhr, "A decision-theoretic model for decentralised query routing in hierarchical peer-to-peer," in *Lecture Notes in Computer Science, Advances in Information Retrieval*. Berlin / Heidelberg: Springer, 2007, pp. 148-159.

[75] C. M. Bowman, P. B. Danzig, D. R. Hard, U. Manber, and M. F. Schwartz, "The Harvest Information Discovery and Access System," *Computer Networks and ISDN Systems*, vol. 28, no. 1-2, pp. 119-125, 1995.

[76] R. J. Bayardo Jr., et al., "InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1997, pp. 195-206.

[77] S. Das, K. Shuster, and C. Wu, "Mobile Agents for Distributed and Heterogeneous Information Retrieval," *Information Retrieval*, vol. 8, no. 3, pp. 383-416, 2005.

[78] R. H. Glitho, "Mobile agents and their use for information retrieval: a brief overview and an elaborate case study," *IEEE Network*, vol. 16, no. 1, pp. 34-41, 2002.

[79] J. Shackleton, D. Cofer, and S. Cooper, "Anytime scheduling for real-time embedded control applications," in *Proceedings of The 23rd Digital Avionics Systems Conference DASC' 04*, vol. 2, Salt Lake City, UT, 2004, pp. 101-110.

[80] E. A. Hansen and Z. R, "Anytime heuristic search," *Journal of Artificial Intelligence Research (JAIR)*, vol. 28, p. 267–297, 2007.

[81] S. Zilberstein, "Using Anytime Algorithms in Intelligent Systems," *AI Magazine*, vol. 17, no. 3, pp. 73-83, 1996.

[82] S. Zilberstein and V. Lesser, "Intelligent Information Gathering Using Decision Models," Computer Science Department, University of Massachusetts, Technical Report 96-35, 1996.

[83] V. Lesser, et al., "BIG: An agent for resource-bounded information gathering and decision making," *Artificial Intelligence*, vol. 118, no. 1-2, pp. 197-244, Apr. 2000.

[84] R. Brooks, T. Arbel, and D. Precup, "Anytime similarity measures for faster alignment," *Computer Vision and Image Understanding*, vol. 110, pp. 378-389, Jun. 2008.

[85] G. E. Moore, "Cramming more Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.

[86] G.-Q. Zhang, G.-Q. Zhang, Q.-F. Yang, S.-Q. Cheng, and T. Zhou, "Evolution of the Internet and its cores," *New Journal of Physics*, vol. 10, no. 12, p. 123027, 2008.

[87] M. Kame and Y. Quintana, "A graph based knowledge retrieval system," in *Proceedings of the 1990 IEEE International Conference on Systems, Man and Cybernetics*, 1990, pp. 269-275.

[88] P. Berkhin, "A Survey on PageRank Computing," *Internet Mathematics*, vol. 2, no. 1, pp. 73-120, 2005.

[89] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 4, pp. 34-43, 2001.

[90] L. Dempsey and R. Heery, "Metadata: A Current View of Practice and Issues," *Journal of Documentation*, vol. 54, no. 2, pp. 145-172, 1998.

[91] J. F. Sowa, "Semantic networks," in *Encyclopedia of Artificial Intelligence*. New York: Wiley, 1992, pp. 1493-1511.

[92] L. Feng, E. Chang, and T. Dillon, "A semantic network-based design methodology for XML documents," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, p. 390–421, Oct. 2002.

[93] Y. Yao, Y. Zeng, N. Zhong, and X. Huang, "Knowledge Retrieval (KR)," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2007, pp. 729-735.

[94] J. Allen, *Natural Language Understanding*, 2nd ed. Addison Wesley, 1994.

[95] H. Nguyen, "Capturing User Intent For Information Retrieval," PhD Thesis, University of Connecticut, 2005.

[96] D. Sleator and D. Temperley, "Parsing english with a link grammar," in *Proceedings of the Third International Workshop on Parsing Technologies*, 1993, pp. 277-292.

[97] M. A. Jones and J. M. Eisner, "A probabilistic parser and its application," in *Statistically-Based Natural Language Processing Techniques: Papers from the 1992 Workshop*, 1992, pp. 20-27.

[98] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[99] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349-1380, 2000.

[100] Y. Rui, T. Huang, and S. Chang, "Image retrieval: current techniques, promising directions and

open issues," *Journal of Visual Communication and Image Representation*, vol. 10, no. 4, pp. 39-62, 1999.

[101] N. S. Vassilieva, "Content-based image retrieval methods," *Programming and Computer Software*, vol. 35, no. 3, pp. 158-180, 2009.

[102] A. Hanbury, "A survey of methods for image annotation.," *Journal of Visual Languages and Computing*, vol. 19, no. 5, pp. 617-627, 2008.

[103] H. Tamura and N. Yokoya, "Image database systems: A survey," *Pattern Recognition*, vol. 17, no. 1, pp. 29-43, 1984.

[104] Y. Tsymbalenko and E. V. Munson, "Using HTML metadata to find relevant images on the Web," in *Proceedings of Internet Computing*, vol. II, 2001, p. 842–848.

[105] E. Hyvönen, S. Saarela, and K. Viljanen, "Ontology Based Image Retrieval," in *Proceedings of WWW*, Budapest, 2003.

[106] B. Arpinar, et al., "Geospatial Ontology Development and Semantic Analytics," *Transactions in GIS*, vol. 10, no. 4, pp. 551-575, 2006.

[107] M. Uschold and M. Grüninge, "Ontologies: Principles, Methods and Applications," *Knowledge Engineering Review*, vol. 11, pp. 93-136, 1996.

[108] R. C. Veltkamp and M. Tanase, "Content-Based Image Retrieval Systems: A Survey," Department of Computing Science, Utrecht University, Technical Report No. UU-CS-2000-34, 2000.

[109] J. Hare, P. H. Lewis, and P. G. B. Enser, "Mind the gap: another look at the problem of the semantic gap in image retrieval," *Proceedings of SPIE*, vol. 6073, p. 607309, 2006.

[110] T. Deselaers, D. Keysers, and H. Ney, "Features for Image Retrieval: A Quantitative Comparison," *Lecture Notes in Computer Science*, vol. 3175, pp. 228-236, 2004.

[111] W.-Y. Ma and H. J. Zhang, "Benchmarking of image features for content-based retrieval," in *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers*, 1998, pp. 253-257.

[112] Y. Wang, T. Mei, S. Gong, and X.-S. Hua, "Combining global, regional and contextual features for automatic image annotation," vol. 42, no. 2, pp. 259-266, 2009.

[113] M. Flickner, et al., "Query by Image and Video Content: The QBIC System," *Computer*, vol. 28, no. 9, pp. 23-32, 1995.

[114] R. M. Rickman and J. Stonham, "Content-based image retrieval using color tuple histograms," in *Proceedings of SPIE*, vol. 2670, 1996.

[115] W. Y. Ma and B. S. Manjunath, "A comparison of wavelet transform features for texture image annotation," in *Proceedings of the International Conference on mage Processing*, vol. 2, 1995, pp. 256-259.

[116] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying.," in *Proceedings of the 22nd Annual Conference on Computer Graphics and interactive Techniques*, New York, 1995, pp. 277-286.

[117] S. Ardizzoni, I. Bartolini, and M. Patel, "Windsurf: Region-Based Image Retrieval Using Wavelets," in *10th International Workshop on Database and Expert Systems Applications*, 1999, p. 167.

[118] A. Natsev, R. Rastogi, and K. Shim, "WALRUS: A Similarity Retrieval Algorithm for Image Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 3, pp. 301-316, 2004.

[119] F. Jing, M. Li, H.-j. Zhang, and B. Zhang, "An Efficient and Effective Region-Based Image Retrieval Framework," vol. 13, no. 5, pp. 699-709, 2004.

[120] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image segmentation using expectation-maximization and its application to image querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, 2002.

[121] J. Li, J. Z. Wang, and G. Wiederhold, "IRM: Integrated Region Matching for Image Retrieval," in *Proceedings of ACM Multimedia Conference*, Los Angeles, CA, 2000, pp. 147-156.

[122] W. Y. Ma and B. Manjunath, "NETRA: A Toolbox for Navigating Large Image Databases," in *Proceedings of IEEE International Conference on Image Processing*, vol. I, Santa Barbara, CA, 1997, p. 568–571.

[123] A. Natsev, "Multimedia Retrieval By Regions, Concepts, and Constraints," Ph.D. Thesis, Department of Computer Science, Duke University, Durham, 2001.

[124] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture Libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947-963, 2001.

[125] H.-K. Lee and Y.-S. Ho, "A Region-Based Image Retrieval System Using Salient Point Extraction and Image Segmentation," in *Advances in Multimedia Information Processing — PCM 2002*. Berlin / Heidelberg: Springer, 2002, pp. 209-216.

[126] A. Natsev, A. Chadha, B. Soetarman, and J. S. Vitter, "CAMEL: Concept Annotated iMagE Libraries," in *Storage and Retrieval for Image and Video Databases, SPIE*, San Jose, CA, 2001.

[127] J. Li and J. Z. Wang, "Real-time Computerized Annotation of Pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 985-1002, 2008.

[128] J. Li and J. Z. Wang, "Real-time Computerized Annotation of Pictures," in *Proceedings of the ACM Multimedia Conference*, Santa Barbara, CA, 2006, pp. 911-920.

[129] M. L. Kherfi, D. Ziou, and A. Bernardi, "Image Retrieval from the World Wide Web: Issues, Techniques, and Systems," *ACM Computing Surveys*, vol. 36, no. 1, pp. 35-67, 2004.

[130] S. Berretti and A. D. Bimbo, "Color Spatial Arrangement for Image Retreival by Visual Similarity," in *Color Image Processing: Methods and Applications*, R. Lukac and K. N. Platoniotis, Eds. 2007, pp. 227-258.

[131] J. Fauqueur and N. Boujemaa, "Region-Based Image Retrieval: Fast Coarse Segmentation and Fine Color Description," *Journal of Visual Languages and Computing (JVLC), Special issue on Visual Information Systems*, vol. 15, 2003.

[132] J. D. Foley, A. v. Dam, and S. K. Feiner, *Computer Graphics: Principles and Practice in C*, 2nd ed. Addison-Wesley Professional, 1995.

[133] K. P. Soman and K. I. Ramachandran, *Insight into Wavelets From Theory to Practice*, 2nd ed. New Delhi: Prentice Hall of India, 2005.

[134] R. Xu and D. Wunsch II, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645-678, May 2005.

[135] P. Berkhin, "A Survey of Clustering Data Mining Techniques," in *Grouping Multidimensional Data*. Berlin Heidelberg: Springer, 2006, pp. 25-71.

[136] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data - An Introduction to Cluster Analysis*. Wiley, 1990, vol. Wiley Series in Probability and Mathematical Statistics.

[137] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A New Data Clustering Algorithm and Its Applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141-182, 1997.

[138] S. Zhong and J. Ghosh, "A unified framework for model-based clustering," *Journal of Machine Learning Research*, vol. 4, pp. 1001-1037, Dec. 2003.

[139] C. Fraley, "Algorithms for model-based Gaussian hierarchical clustering," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, p. 270–281, 1999.

[140] M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain, "Simultaneous Feature Selection and Clustering Using Mixture Models," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1154-1166, 2004.

[141] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *ACM SIGMOD International Conference on Management of Data*, Montreal, 1996.

[142] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA: Morgan Kaufmann Publishers Inc, 1992.

[143] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[144] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the multiple-instance problem with

axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31-71, 1997.

[145] M. Oded, "Learning from ambiguity," PhD Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1998.

[146] O. Maron and A. L. Ratan, "Multiple-instance learning for natural scene classification," in *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, 1998, pp. 341-349.

[147] X. Xu and E. Frank, "Logistic Regression and Boosting for Labeled Bags of Instances," in *Proceedings of the PacificAsia Conference on Knowledge Discovery and Data Mining*, 2004, pp. 272-281.

[148] P. D. Allison, *Logistic Regression Using the SAS System: Theory and Application*, 1st ed. Wiley Publication-SAS Institute, 2001.

[149] D. W. Hosmer and S. Lemeshow, *Applied logistic regression* , 2nd ed. Wiley-Interscience Publication, 2000.

[150] P. Komarek, "Logistic Regression for Data Mining and High-Dimensional Classification," Robotics Institute, Carnegie Mellon University, Pittsburg, PA, Tech. report CMU-RI-TR-04-34, 2004.

[151] C. Perlich, F. Provost, and J. Simonoff, "Tree Induction vs. Logistic Regression: A Learning-curve Analysis," *Journal of Machine Learning Research*, vol. 4, pp. 211-255, 2003.

[152] T. Mitchell. (2005) Carnegie Mellon University Website. [Online]. http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

[153] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1999.

[154] J. Zhang, R. Jin, Y. Yang, and A. G. Hauptmann, "Modified Logistic Regression: An Approximation to SVM and Its Application in Large-Scale Text Categorization," in *Proceedings of the Twentieth International Conference on Machine Learning*, Washington DC, 2003.

[155] R. Ksantini, D. Ziou, B. Colin, and F. Dubeau, "Logistic Regression Models for a Fast CBIR Method Based on Feature Selection," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007.

[156] G. Caenen and E. J. Pauwels, "Logistic regression model for relevance feedback in content based image retrieval," in *Proceedings of SPIE*, vol. 4676, 2002, pp. 49-58.

[157] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[158] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[159] X. Li, "Improving Multi-Agent Coalition Formation in Complex Environment," PhD Thesis, Computer Science, University of Nebraska, Lincoln, Nebraska, 2007.

[160] M. P. Georgeff and A. L. Lansky, "Reactive reasoning and planning," in *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87)*, Seattle, 1987, pp. 677-682.

[161] D. S. Weld, "Recent advances in AI planning," *AI Magazine*, vol. 20, no. 2, p. 93–123, 1999.

[162] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[163] R. Bellman, "On the Theory of Dynamic Programming," *Proceedings of the National Academy of Science*, vol. 38, no. 8, p. 716–719, Aug. 1952.

[164] W. S. Lovejoy, "A survey of algorithmic methods for partially observable Markov decision processes," *Annals of Operations Research*, vol. 28, no. 1, pp. 47-65, 1991.

[165] G. E. Monahan, "A survey of partially observable Markov decision Processes:Theory, models, and algorithms," *Management Science*, vol. 28, no. 1, pp. 1-16, 1982.

[166] P. Poupart, "Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes," PhD Thesis, Graduate Department of Computer Science, University of Toronto, 2005.

[167] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[168] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: A critical Survey," Stanford University, Technical Report, 2003.

[169] C. H. Papadimitriou and J. N. Tsitsiklis., "The complexity of Markov decision," *Mathematics of Operations Research*, vol. 12, no. 3, p. 441–450, 1987.

[170] C. Boutilier, T. Dean, and S. Hanks, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *Journal of Artificial Intelligence Research*, pp. 1-94, 1999.

[171] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored MDPs," in *14th Neural Information Processing Systems (NIPS 14)*, Vancouver, Canada, 2001, pp. 1523-1530.

[172] T. Dean, R. Givan, and S. Leach, "Model Reduction Techniques for Computing Approximately Optimal Solutions for Markov Decision Processes," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997, pp. 124-131.

[173] Z. Feng and E. A. Hansen, "Approximate planning for factored POMDPs," in *Proceedings of the Sixth European Conference on Planning*, Toledo, Spain, 2001.

[174] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663-704, 2008.

[175] M. Hauskrecht, "Value-Function Approximations for Partially Observable Markov Decision Processes," *Journal of Artificial Intelligence Research*, pp. 13-94, 2000.

[176] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-04)*, 2004, p. 520–527.

[177] T. Smith and R. Simmons, "Point-based POMDP algorithms: improved analysis and implementation," in *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 2005, p. 542–547.

[178] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: an anytime algorithm for POMDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003, p. 1025–1032.

[179] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: scaling up," in , 1995, p. 362–370.

[180] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 3, pp. 297-331, Dec. 2009.

[181] M. E. desJardins, E. H. Durfee, C. L. Ortiz Jr., and M. J. Wolverton, "A Survey of Research in Distributed, Continual Planning," *AI Magazine*, vol. 20, no. 4, pp. 13-22, 1999.

[182] R. Jensen and M. M. Veloso, "Interleaving Deliberative and Reactive Planning in Dynamic Multi-Agent Domains," in *Proceedings of the AAAI Fall Symposium on Integrated Planning for Autonomous Agent Architectures*, 1998.

[183] O. Sapena and E. Onaindía, "Planning in highly dynamic environments: an anytime approach for planning under time constraints," *Applied Intelligence*, vol. 29, no. 1, pp. 90-109, Aug. 2008.

[184] D. N. Kinny and M. P. Georgeff, "Commitment and effectiveness of situated agents," in *Proceedings of the 12th international Joint Conference on Artificial intelligence*, vol. 2, Sydney, Australia, 1991, pp. 82-88.

[185] S. Koenig and X. Sun, "Comparing real-time and incremental heuristic search for real-time situated agents," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 3, pp. 313-341, 2009.

[186] A. S. Rao and M. P. Georgeff, "BDI-agents: from theory to practice," in *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, 1995.

[187] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

[188] A. E. Eiben and C. A. Schippers, "On Evolutionary Exploration and Exploitation," *FUNDAMENTA INFORMATICAE*, vol. 35, no. 1/4, pp. 35-50, 1998.

[189] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 4th ed. John Wiley & Sons, 2006.

[190] E. Santos Jr. and E. S. Santos, "A framework for building knowledge-bases under uncertainty," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 11, no. 2, pp. 265-286, 1999.

[191] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 1st ed.

Morgan Kaufmann, 1988.

[192] G. Box, S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd ed. Wiley-Interscience, 2005.

[193] G. W. Cobb, *Introduction to Design and Analysis of Experiments*. Springer, 1998.

[194] Bredin, et al., "A game-theoretic formulation of multi-agent resource allocation," in *Proceedings of the Fourth international Conference on Autonomous Agents*, Barcelona, Spain, 2000, pp. 349-356.

[195] E. E. Santos, L. Pan, D. Arendt, and M. Pittkin, "An Effective Anytime Anywhere Parallel Approach for Centrality Measurements in Social Network Analysis," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, 2006, pp. 4693-4698.

[196] L. Pan and E. E. Santos, "An anytime-anywhere approach for maximal clique enumeration in social network analysis," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* , 2008, pp. 3529-3535.