Ministry of Higher Education and
Scientific Research University of Kasdi
Merbah Ouargla Faculty of New
Technologies of Information and
Communication Department of
Computer Science and Information
Technologies

ACADEMIC MASTER

Domain: Mathematics and Computer Science

Faculty: Computer Science

Specialty: Industrial Computing

PARALLEL METAHEURISTIC FOR SEMANTIC BASED QUERY EXPANSION

Evaluation date:
18/06/2023

Mrs BENKHROUROU CHAFIKA	**President**	**UKM Ouargla**
Mr.CHRIET ABDELHAKIM	**Examiner**	**UKM Ouargla**
Mr BEKKARI FOUAD	**Supervising**	**UKM Ouargla**

Presented by:
AYACHI NOUR ELHOUDA
GHOULA HADJER

Year University: 2022/2023

Acknowledgements

First and foremost, praises and thanks to Allah, the almighty, for his showers of blessings throughout our research and its successful completion.

We would like to acknowledge and give our warmest thanks to our supervisor Mr.Bekkari Fouad who made this work possible, his guidance and advice carried us through all stages of writing our project. We would like to thank Mrs.Benkhrourou Chafika for having honored us by chairing the defense jury, and Mr.Chriet Abd Elhakim for accepting to examine this work.

We would like also to give special thanks to our family and friends as a whole for their continuous support and understanding when undertaking our research and writing our project, your prayers for us were what sustained us this far.

Abstract

Word mismatch problem between the user's query and the retrieved search results is one of the biggest problems facing the information retrieval(IR) field. Due to this problem, several techniques have been proposed such as query expansion(QE), which improves the IR performance by giving a more suitable extended query for users in comparison to the original query. In this work, we are looking for the best combination of the words in the extended queries using the semantic-based query expansion approach "ConceptNet". To find these combinations we applied a low-level parallel metaheuristic Iterated local search(ILS) and a high-level parallel metaheuristic Accelerated particle swarm optimization(APSO) with Local search.

Keywords: Information retrieval, Query expansion, ConceptNet, Iterated local search(ILS),Accelerated particle swarm optimization(APSO), Parallel metaheuristic.

ملخص

تعد مشكلة عدم تطابق الكلمات بين استعلام المستخدم ونتائج البحث المسترجعة واحدة من أكبر المشكلات التي تواجه مجال استرداد المعلومات. بسبب هذه المشكلة، تم اقتراح العديد من التقنيات مثل توسيع الاستعلام ، مما يحسن أداء استرجاع المعلومات من خلال إعطاء استعلام موسع أكثر ملاءمة للمستخدمين مقارنة بالاستعلام الأصلي. في هذا العمل، نحن نبحث عن أفضل مزيج من الكلمات الواردة في الاستفسارات الموسعة باستخدام نهج توسيع الاستعلام القائم على الدلالة آكونسبت نات». للعثور على هذه التركيبات، قمنا بتطبيق ميتاهوريستيك متوازي منخفض المستوى المتمثل في البحث المحلي المتكرر و ميتاهوريستيك عالي المستوى المتمثل في تحسين سرب الجسيمات المتسارع المتوازي عالي المستوى مع البحث المحلي.

الكلمات المفتاحية : استرجاع الاستعلام، تمديد الاستعلام، البحث المحلي المتكرر، تحسين سرب الجسيمات

Résumé

Le problème de correspondance de mots entre la requête de l'utilisateur et les résultats de recherche récupérés est l'un des plus grands problèmes auxquels fait face le champ de recherche d'information(RI). En raison de ce problème, plusieurs techniques ont été proposées telles que l'expansion de requête (ER), qui améliore la performance IR en donnant une requête étendue plus appropriée pour les utilisateurs par rapport à la requête originale. Dans ce travail, nous recherchons la meilleure combinaison de les mots dans les requêtes étendues en utilisant l'approche d'expansion de requête sémantique ConceptNet. Pour trouver ces combinaisons, nous avons appliqué une métaheuristique parallèle de bas niveau recherche locale itérée(ILS) et une métaheuristique parallèle de haut niveau Optimisation accélérée des essaims de particules(APSO) avec le Recherche Locale.

Les mots clés: Recherche d'informations, expansion de la requête, ConceptNet, recherche locale itérée (ILS), optimisation accélérée des essaims de particules (APSO), métaheuristique parallèle.

Contents

List of Figures

General Introduction

Nowadays, informations are easily accessible on the Internet, and we can access it whenever we want. With the growth and availability of information, it became challenging to store, organize, and retrieve them. An ever-increasing volume of data has made classical information retrieval systems suffer from the problem of incompatibility between retrieved documents and the user's query. To remedy this problem, many techniques have emerged, such as query expansion, that aim to enhance the effectiveness of information retrieval systems.

Information retrieval is the process of retrieving documents that are related to a query entered by the user from a large amount of data. Getting the appropriate information that is compatible with what the user wants in a better and faster has therefore become an essential problem in this field due to the abundance and diversity of information available on the web.

Due to this problem, query expansion has been proposed to solve it. Query expansion technique is one of the promising approaches to improve the performance and reliability of information retrieval system, which means adding one or more suitable expansion keywords in comparison to the initial queries given by the user, the more appropriate the added words are, the greater the opportunity to retrieve the documents that are related to the initial query, query expansion strives to improve recall and precision, leading to more accurate search results.

Although query expansion helps in improving the performance of information retrieval, it's not enough. The word-mismatch problem between what the user wants and what the system offers him is one of the biggest problems facing this field. So the researchers used other techniques to improve the performance of query expansion.

Metaheuristics are techniques that have shown their effectiveness in solving various complex problems in the least amount of time and at the lowest cost, such as Particle Swarm Optimization(PSO), Genetic Algorithm(GA), Fireworks Algorithm(FWA), and Iterated Local Search(ILS).

We will deal with the problem of query expansion as a combinatorial problem using two algorithms "Accelerated particle swarm optimization (APSO) with Local Search " using MPI implementation and "Iterated local search (ILS)" with GPU implementation to solve this problem, each one of them proved their effectiveness

in solving many other problems. In this work, we have chosen to use the semantic-based query approach "ConceptNet". ConceptNet has been widely used in document searches for identifying indexing terms, that have similar concepts to the user's query, these terms can be used therefore to augment the initial query.

Our memory is organized into three chapters, beginning with the general introduction and ending with a general conclusion.

The three chapters are titled respectively and detailed as follows:

In the first chapter we draw a state of the art, the first section aim to present the field of IR we start by introducing it and showing its importance, then we talk about IR system and their models and gave some examples of IR applications.

Then in the second section we introduce Query Expansion, starting with its definition and models, then we talked about the QE process and its approaches, and we finished it with the Semantic Query expansion.

In the second chapter, we divide it into three sections, the first section is about metaheuristics, and we looked at its definition and classification. In the second chapter, we talked about hybridization's definition and classification.

The last section includes parallelism architecture, and also we define GPU, MPI, parallel metaheuristic, and parallel classification.

Finally, in the third chapter, we presented our experimental environment, the used tools, the implementation stage of the system, and presents the obtained results as well as the analysis of the results.

Motivation

Our inspiration for this work was the existence of parallel machines that would help us to improve our approach, the existence of the programming technology such as GPU and MPI, that help us to develop parallel applications that would be able to execute in parallel and exploit more search spaces.

the strength of metaheuristics in solving many combinatorial problems was an inspiration for this work too. For high-dimensional problems, they become limited in terms of effectiveness and runtime, that's why we want to use two techniques which are parallelization, which will help us to reduce execution time, and hybridization which allow us to obtain better high-quality solutions.

In this work we used Iterated local search(ILS) and Accelerated particle swarm optimization(APSO) with local search, these metaheuristics proved their effectiveness in solving many difficult problems and gave good results.

Chapter 1

State Of The Art

1.1 Introduction

People have known how crucial it is to archive and find information for thousands of years. With the development of computers, it became possible to store vast amounts of information; and obtaining valuable information from such collections became an essential need; the field of Information Retrieval(IR) arose out of this necessity in the 1950s.

At this time, there is a huge amount of data available on the internet and it is growing exponentially, finding the right information that we are needing became a challenge because of the huge diversity and availability of information on the web, we still struggle to find information due to several reasons such as; the retrieved search results are not that we are looking for, or the search queries are too short because when searching we only enter the least possible of the word(average size of a web search is 2.4 words). To overcome this problem, query expansion has been proposed.

In this chapter, we started by giving a brief introduction then we defined Information retrieval, its components, and the IR system and give some of IR applications. In the last section of this chapter, we defined QE, mentioning its models and process, then we talked about its approaches

We conclude this chapter with a conclusion, we mentioned the problem we are dealing with and propose a solution which will be treated as an introduction for the second chapter

1.2 Information Retrieval

1.2.1 Information Retrieval Definition

Information retrieval is mainly seen as a branch of computer science concerned with the representation, storage and access of information, and it has pertained to the structuring and retrieval of information from massive database sources.[1]

The goal of information retrieval (IR) is to find results that are relevant to the user's needs in response to a user's query.[2] The challenge is how to achieve a good match between these two in order to ensure that the information presented is relevant to the user who made the original query. This process involves several stages

beginning with representing data and concluding with returning relevant information to the user.[1]

1.2.2 Information Retrieval Components

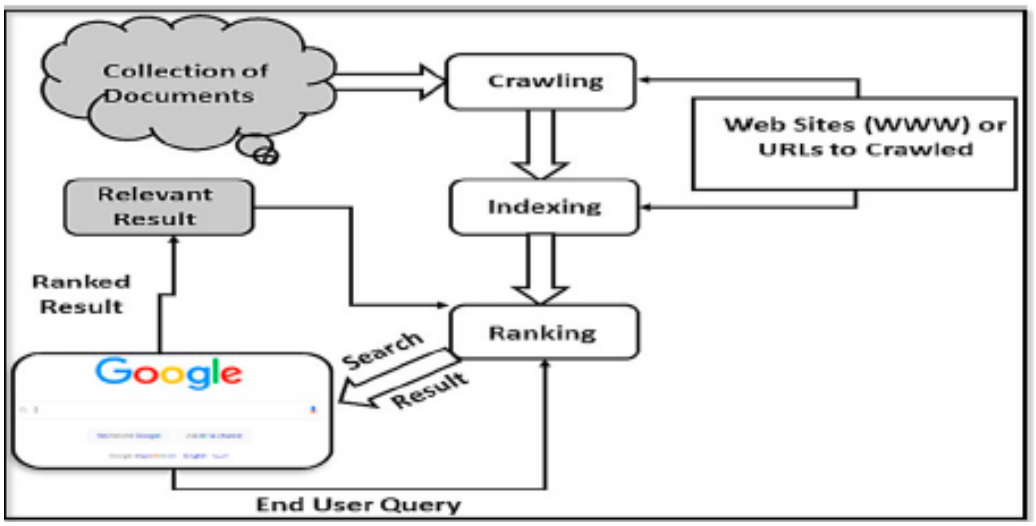

Figure 1.1: Informatin retrieval components

The first component, as seen in figure , is crawling. The crawler component searches for and retrieves documents for the search engine. Crawlers come in a number of shapes and sizes, but the most common is the standard web crawler, which follows connections on internet pages to find and download new pages. As a beginning point for site search, a web crawler can be restricted to a specific site, such as a college.[2]

The second component, known as indexing, is the technique for representing documents. It essentially means that the system produces a document index. As a result, the query representation process is discovered. During this stage, the user creates a query to retrieve relevant information. Following that, the system searches the index for pages relevant to the query and presents them to the user, which is what we call ranking. The final step is for users to submit relevant feedback to the search engine. [2]

1.2.3 Information retrieval system

An information retrieval (IR) system is a collection of algorithms that makes it easier for presented documents to be relevant to search queries, which are generally text documents but may also include multimedia. Simply said, it helps users discover the information they need by sorting and ranking documents according to their search terms, there are many examples like Google, Bing, Yahoo...[3]

To "find relevant information or a document that satisfies user information needs" is the primary objective of information retrieval systems (IRS), the procedures that IRSs typically use to accomplish this purpose are as follows:[1]

1. During the indexing process, document contents are summarized.
2. All stop words and common words are eliminated throughout the filtering process.
3. The core function of IRS is searching, there are several methods for finding documents that suit users' needs.

There are three basic processes an information retrieval system has to support: the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations.

Representing the documents is usually called the indexing process. The process takes place offline, that is, the end user of the information retrieval system is not directly involved. The indexing process results in a representation of the document. The process of representing their information need is often referred to as the query formulation process. The resulting representation is the query. Comparing the two representations is known as the matching process. Retrieval of documents is the result of this process.[1]

Indexing stage

An essential step in Information Retrieval (IR) systems is indexing. Since it is the first stage in IR and helps in effective information retrieval, it forms the core functioning of the IR process. The documents are reduced to their informative terms through indexing. It offers a mapping of the terms to the corresponding documents where they are used.[4]

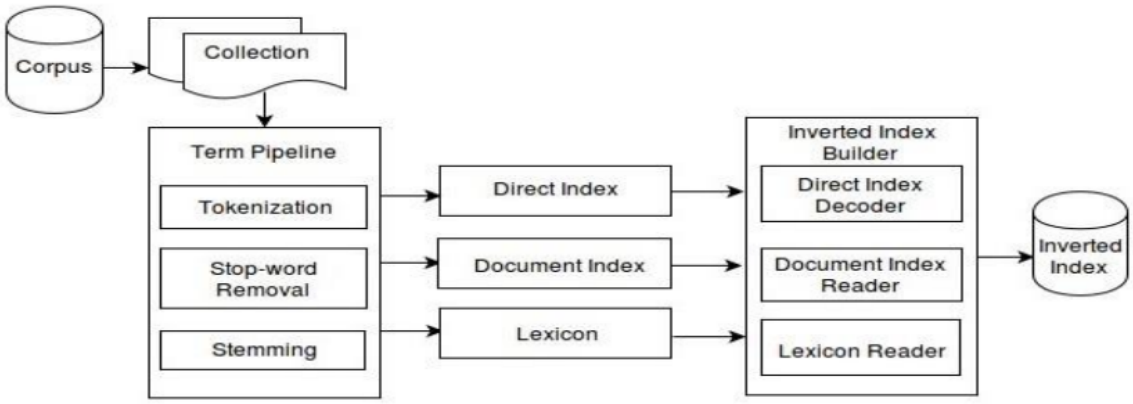

Figure 1.2: Indexing process

The four stages of indexing include *content specification*, *tokenization of documents*, *processing of document terms*, and *index construction*. The index can be kept in the form of many data structures, including direct indexes, document indexes, lexicons, and inverted indexes.[5]

Search stage

Searching is defined as the process of comparing a query to all documents in a database in order to retrieve information related to that query. It is difficult to extract relevant information from documents; therefore, these documents must first be properly represented using any IR model.

An IR model explains how the document representation, user query representation, and retrieval mechanism or process are elaborated, There are three fundamental IR models: Boolean model, vector-space model, and probabilistic model.

Boolean model

The Boolean model was the first information retrieval model and is likely the one that has received the most criticism[1]. It is extensively used in search engines because this simple yet fast model performs better in searching techniques that emphasize speed over precision.[6]

The Boolean model uses set theory, or boolean algebra and its three components, AND, OR, and NOT, for the formulation of queries, but it has a major weakness:

it fails to rank the list of documents that are retrieved as results, according to the Boolean model, each document is linked to a certain set of keywords and users. Queries can also be represented by keyword expressions separated by AND, OR, or NOT, the boolean model's retrieval function classifies a document as either relevant or irrelevant.[6]

Probabilistic Model

The most important or fundamental function of the probabilistic model is its ability to rank documents by their probability to be relevant given a user's query, both documents and user queries are represented by binary vectors, with each binary vector component indicating whether a specific document attribute/component or term is in the document or query or not. The index term weight variables for the probabilistic model are entirely binary rather than using probabilities.[6]

Vector space model

A model based on Luhn's similarity criterion that has a stronger theoretical motivation was proposed by Gerard Salton and his colleagues. They viewed the query and index representations as vectors embedded in a highly dimensional Euclidean space, where each term is given its own dimension.[1]

As indicated by the name, this model links terms to documents through the use of vectors. The vector model is a straightforward model that uses vectors to indicate the direction in which documents are relevant to the terms. The model was proposed in response to the Boolean Model because it uses binary values to evaluate relevancy, which was seen to be ambiguous when obtaining documents.[6]

Each query and each document have a vector connected with them, thus we can say that query Q and document D each have a vector of the form Q=w1,w2,w3,...wN and D=w1,w2,w3,...wN. Documents and queries are represented as vectors in the Vector Space Model (VSM), and the angle between the two vectors is calculated using the similarity cosine function.

The term-weighting system known as tf-idf weighting has been introduced for the Vector Space Model. These weights include a term frequency (tf) factor that measures the frequency of occurrence of terms in document or query texts and an inverse document frequency(idf) factor that measures the inverse of the number of documents that contain a query or document term.[6]

The vector model has advantages over the other models since it uses term weights rather than binary values, which increases the document's relevancy. Partial matching is done, as well as ranking the documents based on their relevancy.

1.2.4 Information Retrieval Applications

Systems for information retrieval (IR) were initially developed to help with the management of huge amounts of data. Many universities, organizations, and public libraries now provide access to books, documents, journals, and other types of data through the use of IR systems[2]. Information retrieval is used in many different applications nowadays. The following are some examples of IR system applications:

Digital Library: A digital library is one that uses computers to access collections that were stored digitally. Through computer networks, digital content can be accessed remotely or locally, A digital library is a sort of IR system.

Search Engines: One of the most useful information retrieval strategies for large-scale text collections is a search engine. Web search engines are the most famous examples, but there are also federated search, desktop search, mobile search, enterprise search, web search, and social search.

Multimedia Search: In order to obtain information that is different from textual search, this type of search can be applied by multi-modal search interfaces, which include other types of media as well[1]. As an example, an image retrieval system is a multimedia search system in a computer that allows users to browse, search for, and retrieve images from huge collections of digital images.

1.3 Query Expansion

1.3.1 Query Expansion Definition :

Query expansion is one of the promising approaches to dealing with the word mismatch problem in information retrieval (IR). The main motivation of query expansion is to expand a user's query by adding meaningful terms that are related to the original query terms from other documents in the corpus or by adding synonyms from a thesaurus; the added words should be very effective and improve retrieval performance. Adding related words to the initial query can improve the number of relevant documents identified, thereby increasing the probability of relevant document discovery.

1.3.2 Query Expansion Models

Query expansion models (QE) are techniques used to increase the effectiveness of information retrieval by adding new terms. QE models fall into the following categories according to automation and end-user involvement:

Manual Query Expansion :

Manual query expansion encourages and motivates the user to refine the original query through heuristics by providing a list of potential terms from the query log and allowing them to select the most appropriate terms.

Automatic Query Expansion :

AQE automatically expands search queries by analyzing documents returned from first-pass retrieval. It weights candidate terms for expansion and expands the original query as needed.

Interactive Query Expansion :

Query reformulation is achieved through joint cooperation between the system and the user. It is a human-in-the-loop approach where the system returns search results and the user chooses meaningful results.[7]

1.3.3 Query Expansion Process :

The process of generating query expansion consists of mainly four steps: preprocessing of data sources, term weights and ranking, term selection and query reformulation.

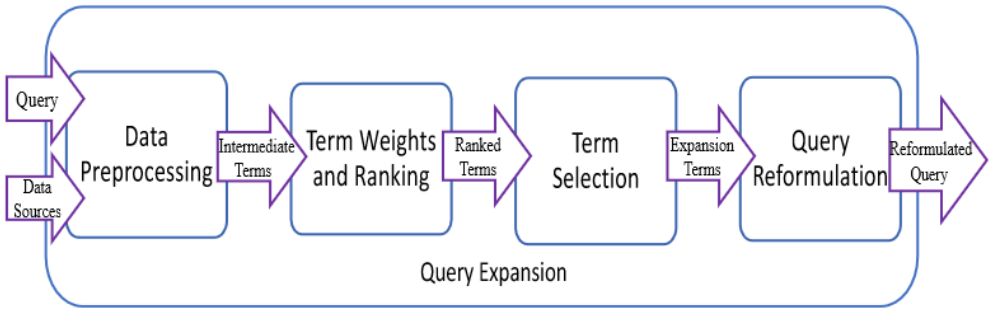

Figure 1.3: Query expansion working model

Preprocessing of Data Sources :

Preprocessing data sources aims to extract terms that will enhance the user's initial query depending on the data sources and appoaches used, it consists of the following four sub-steps:[7]

1. Text extraction from data sources(extraction of the whole texts from the specific data source used for query expansion).
2. Tokenization (process of splitting the stream of texts into words).
3. Stop word removal (removal of frequently used words e.g., articles, adjective, prepositions, etc.).
4. Word stemming (process for reduction of derived or inflected words to their base word).

Weighting and Ranking of Query Expansion Terms :

QE involves assigning weights and ranks to query expansion terms based on the user's query and texts obtained from data sources. These weights indicate the relevance of the terms and are used to rank documents based on relevancy. There are various techniques for ranking and weighting query expansion terms. Researchers classifies the techniques into four categories on the basis of relationship between the query terms and the expansion features.[7]

One-to-One Association: Such as WordNet to find synonyms and similar terms for the query terms.

One-to-Many Association: Correlates one query term to many expanded query terms.

Feature Distribution of Top Ranked Documents: Deals with the top retrieved documents from the initial query and considers the top weighted terms from these documents.

Query Language Modeling: Constructs a statistical model for the query and choses the expansion terms having highest probability.

Selection of Query Expansion Terms:

The previous section dealt with the ranking and weighting of expansion terms. Following this process, the top-ranked terms are chosen for query expansion. The selection of terms is done on an individual basis; mutual dependency of terms is not taken into consideration. The independence assumption might be questionable, but certain experimental studies suggest that it might be empirically equitable. It is possible that the chosen query technique generates a large number of expansion terms, but it may not be realistic to employ them all; as a result, noise reduction only allows for the selection of a limited number of expansion terms.[7]

Query Reformulation:

In the last step of query expansion, the expanded query is reformulated to achieve better results when used for retrieving relevant documents. The reformulation is done based on weights assigned to the individual terms of the expanded query known as query reweighting.[7]

1.3.4 Query Expansion Approaches

Query expansion approaches can be classified mainly into two categories based on global analysis, which obtains expansion terms on the statistics of terms in the whole corpus, and local analysis, which extracts expansion terms from a subset of the search results.

Global Analysis :

One of the earliest techniques to produce consistent and effective improvements through query expansion was global analysis. The main idea behind global analysis is to evaluate a term's context to determine its similarity to other terms. Each term is

allocated a weight, and expansion terms can be given less weight than the initial query terms. Global analysis, different from local analysis, selects expansion terms based on information from the entire document collection. To select the relevant terms for a given query, global analysis usually depends on a set of statistical relationships.[8]

Usually, global analysis relies on a set of statistical relationships to select relevant terms for a given query.

Global analysis can be classified into four categories on the basis of query terms and data sources:

Linguistic-based Approaches: Methodologies in this category examine lexical, morphological, semantic, and syntactic term relationships to reformulate or extend query terms, using thesaurus, dictionaries, ontologies, Linked Open Data (LOD) cloud, WordNet, and other knowledge resources.[7]

Corpus-based Approaches: Corpus-based Approaches look at the entire text corpus's content to identify the expansion features that are used for query expansion. They use co-occurrence statistics to link terms together to build sentences, paragraphs, or clusters of words for query expansion.[7]

Search log-based Approaches: Search log analysis is used to analyze user input, which is a key source for query growth due to the expanding size of the web and the rising use of web search engines. User feedback is used to provide a set of related terms depending on the user's initial query.[7]

Web-based Approaches: These strategies, which have recently gained popularity, include anchor texts and Wikipedia to broaden the user's original inquiry. Anchor text serves as a brief description of a page, while Wikipedia is the largest free online encyclopedia and is continuously updated, making it the perfect information source for query expansion.[7]

To expand queries, the global analysis generally necessitates corpus-wide statistics and a global association thesaurus. However, it only gives an incomplete solution to the term mismatch problem because it concentrates entirely on the document side and ignores the query side.

Local Analysis :

Local analysis differs from global analysis in that it focuses on a particular subset of documents returned by a query. It is divided into two categories: approaches based on user feedback information and approaches based on information derived from a subset of the returned documents [8]

Relevance Feedback: Rocchio's method was the first to use relevance feedback, which is a collection of user feedback about documents retrieved in response to an initial query. It can be categorized into two types: explicit feedback and implicit feedback. Explicit feedback evaluates the relevance of retrieved documents, while implicit feedback uses user activity to infer user preferences. Relevance feedback suffers from a lack of semantics in the corpus, which limits its applications when the query concept is as general as a disjunction of more specific concepts.[7]

Psuedo-Relevance Feedback: Local feedback, also known as blind feedback or pseudo-feedback, is commonly used in research to overcome the difficulty due to a lack of sufficient relevance judgments. It mimics relevance feedback by assuming the top-ranked documents to be relevant. The idea of local analysis can be traced back to a 1977 paper[Attar and Fraenkel,1977], which proposed the top-ranked documents for a query as a source of information for building an automatic thesaurus. Terms in these documents were clustered and treated as quasi-synonyms.[9]

1.3.5 Semantic Query Expansion

To improve the IR system interpretation of the search query, QE is a technique used to compute expansion terms that are important to the user intent and add them to the initial search query.

The search query is expanded using a corpus in traditional QE methods like global analysis and local analysis, however, semantic QE approaches do not have such limitations as they are built on corpus-independent knowledge structures (e.g., a lexical thesaurus or ontology), the expansion terms are used to broaden the search query with relevant terms (concepts) that are closer to the user's intent, they are derived based on a similarity measure between the initial search query terms and the concepts of the knowledge structure.[10]

As compared to other QE approaches like relevance feedback, where the expansion process depends on the initial search results, semantic QE always has access to the knowledge structure in the expansion mechanism.

To illustrate the value of knowledge structures in the QE process, search query terms were manually disambiguated and obtained from an ontology or thesaurus, a knowledge structure is made up of relationships among concepts, from which the context (semantics) of a concept can be used to derive meaningful expansion terms. [11]

Efthimiadis divided corpus-independent knowledge structures into three categories: dictionaries like the Collins dictionary, general thesaurus like WordNet which are not limited to a specified domain, and domain-specific thesaurus that provide synonyms or other related relationships between concepts of a domain.[10]

Semantic Query Expansion Approaches

Semantic-based QE approaches are classified into three types: the thesaurus-based approach, the Ontology-based approach, and word-embedding-based semantic query expansion.

Thesaurus Based Approach: The thesaurus-based approach is a widespread technique used to select appropriate terms for query enhancement like WordNet. In this approach, the query is reformulated using synonyms, hyponyms, and other relationships based on the context of the query.[12]

Word Embedding Based Approach : Contextual data responds well to this approach. Instead of searching for synonyms in external data sources, it helps contextually derive the meaning of the user's given query.[12]

Ontologie Based Approach : Ontology is used to convert implicit knowledge to explicit knowledge, allowing users to access and share that field knowledge. Ontology approaches are classified into two categories: Domain-specific, which belong to the field of agriculture, medicine, business, law, and sports and domain-independent, which consist of general purpose vocabulary, concepts, and instances.Candidate expansion terms are retrieved based on the similarity between query terms and ontology concepts. [12]

ConceptNet :

ConceptNet is one of the largest common-sense knowledge base that covers semantic relationships between real-world concepts. Similar to Wikipedia, ConceptNet reflects the "knowledge of the crowds" and was built by assembling huge number of sentences that serve as assertions about the real world from various online contributors.[13]

Semantic networks are used by ConceptNet as a model for knowledge representation, the knowledge in ConceptNet is gathered from a variety of resources, including crowd-sourced resources like Open Mind Common Sense and Wiktionary, games with a purpose such as Verbosity, and expert-created resources like Wordnet. Its semantic network's nodes stand in for semi-structured natural language fragments (eg, "food," "grocery shop," "buy food," and "at home") and represent real world concepts. A semantic connection between two concepts is represented by an edge connecting two nodes.[14]

In contrast to ontologies such as WordNet, ConceptNet is not limited to hyponym/hypernym relations and offers a more varied relational ontology with twenty relationship kinds, including causal, spatial, and functional relationships. The network structure of ConceptNet does not require any additional analysis to establish the relations between the concepts, in contrast to online encyclopedias like Wikipedia.[14]

ConceptNet has been demonstrated to be a valuable resource that could enhance retrieval performance, especially for challenging queries.[15]

1.4 Problematic and Proposition

Most of the query expansion techniques work to find the best way to make the added word more valuable for the user to get his satisfaction, in this work we are using the semantic query expansion approach ConceptNet, which enhances the original query by adding more expanded words that are related to the initial query by their concept.

In classical "ConceptNet", they usually pick the top ranked concept but in this work, we are not interested in the top-ranked concept, we are looking for the best combination of n-words from a bag of words but we found that the evaluation phase

was too expensive and take a lot of execution time that's why we will apply a low-level and high-level parallel metaheuristics to reduce it.

1.5 Conclusion

In this chapter, we provide an introduction to the field of information retrieval from the definition and components to concepts related to the methods and structure of its work, and we moved to the importance which is the query expansion, giving a definition and touching on the existing approaches to use which we will use semantic query expansion approach conceptnet in our work.

In the second chapter, we will look at the Iterated Local Search and Accelerated Particle Swarm Optimization metaheuristics and how they work.

Chapter 2

Metaheuristic and Parallelization

2.1 Introduction

In the process of solving challenging optimization issues, metaheuristics have been showing interesting results; however, for high-dimensional problems, they become limited in terms of effectiveness and runtime. Parallel computing appears to be an attractive choice for reducing execution time and improving solution quality due to the independence of metaheuristic components.[16]

By exploiting the increasing performance and programmability of graphics processing units (GPUs) to this aim, GPU-based parallel metaheuristics have been implemented using different designs. Recent research indicates that GPUs are effective co-processors for leveraging complex optimization problems.

2.2 Metaheuristic

2.2.1 Metaheuristic Definition

In iterative master processes, metaheuristic algorithms direct and modify subordinate heuristics to high-quality solutions, they modify either a complete (or partial) single solution at each iteration or a set of such solutions. These algorithms are simple, adaptable, and capable of avoiding local optima because they are derivative-free approaches, as they don't require any knowledge about the gradient of the objective function to determine the global solution, metaheuristic optimization algorithms have been suggested as effective alternatives to traditional deterministic methods.[17]

Metaheuristic algorithms display stochastic behavior; they begin the optimization process by producing random solutions. The primary characteristic of metaheuristic algorithms is their exceptional ability to prevent the algorithms' premature convergence; because of the stochastic nature of algorithms, the techniques operate as a black box, avoiding local optima and efficiently and effectively exploring the search space.[18]

The metaheuristic algorithms trade-off between two of its most important components, exploitation, and exploration. The algorithms completely analyze the promising search space in the exploration phase, and the exploitation phase involves the

local search of any promising area(s) that were discovered in the exploration phase. [18]

2.2.2 Exploration and Exploitation

The ability of an optimization algorithm to "explore" and "exploit" data is the most crucial factor affecting performance. Exploration refers to a search algorithm's ability to explore various areas in the search space to boost the probability to discover a good optimum; Exploitation, on the other hand, refers to the ability to concentrate the search around a promising region to refine a candidate solution.[19]

A good optimization algorithm should optimally balance the two conflicted objectives, which indicates that the ability of exploration and the ability of exploitation should be adjusted via the population diversity analysis when solving different problems or on different search stages. For example, strong exploration ability means that the algorithm has a great possibility to "jump out" of local optima.[19]

2.2.3 Metaheuristic Classification

Metaheuristic algorithms classify into the following two main categories :

Single solution based metaheuristic algorithms

These methods begin their optimization process with a single solution, and during the iterations, their solution is updated; it could result in trapping into local optima and also does not explore the search space thoroughly.[18]

Tabu search : The purpose of Tabu Search is to keep track of the search paths that the search procedure has already visited to avoid it from falling into a local optimum; this may cause the algorithm to accept some inferior solutions to avoid revisiting past paths to get the best solution through a more globalized search.[20]

The visited search paths (forbidden solutions) are stored in a tabu list, which is maintained by a forbidding strategy that determines which solutions are candidates and should be preserved in the tabu list.

Simulated annealing : The Simulated Annealing (SA), was introduced by Kirkpatrick; it is a single solution based metaheuristic optimization method that

was motivated by the metallurgical process of annealing. In the SA, the cooling process of metals is regulated to build stable crystal structures that reduce defects; the SA considers the energy of a thermodynamic system as the values that will be optimized.[21]

The SA is an iterative process that uses a random walk to modify the positions of the candidate solutions; here are considered the changes of temperate, when the temperatures are high the SA operators perform large movements and it has more probabilities to accept solutions far away from the global optimal; however, this situation permits to avoid suboptimal solutions, meanwhile, if the temperatures are low the movement of solutions and the probability to accept bad solutions is low.[21]

Population (multiple) solution based metaheuristic algorithms

These algorithms start their optimization process by generating a population of solutions, with each generation or iteration the population of solutions updates. The algorithms help prevent local optima since they have a great search space exploration and various solutions that work together to help one another; also, they possess the ability to jump to the promising area of the search space. Hence, the majority of difficulties in the real world are solved using population-based algorithms.[18]

Ant colony : The ant colony algorithm simulates the "exploring" and "using" of pheromones by natural ants; in other words, ants use the pheromones left by other ants in the foraging process, choose a path or explore a new path with a certain probability, and secrete pheromones at the same time.

The pheromone on the path will accumulate, volatilize and spread, and affect the following ants, ants tend to choose a path with high pheromone concentration and eventually search for the optimal path; path selection and pheromone update are the two most crucial elements in the algorithm's implementation.[22]

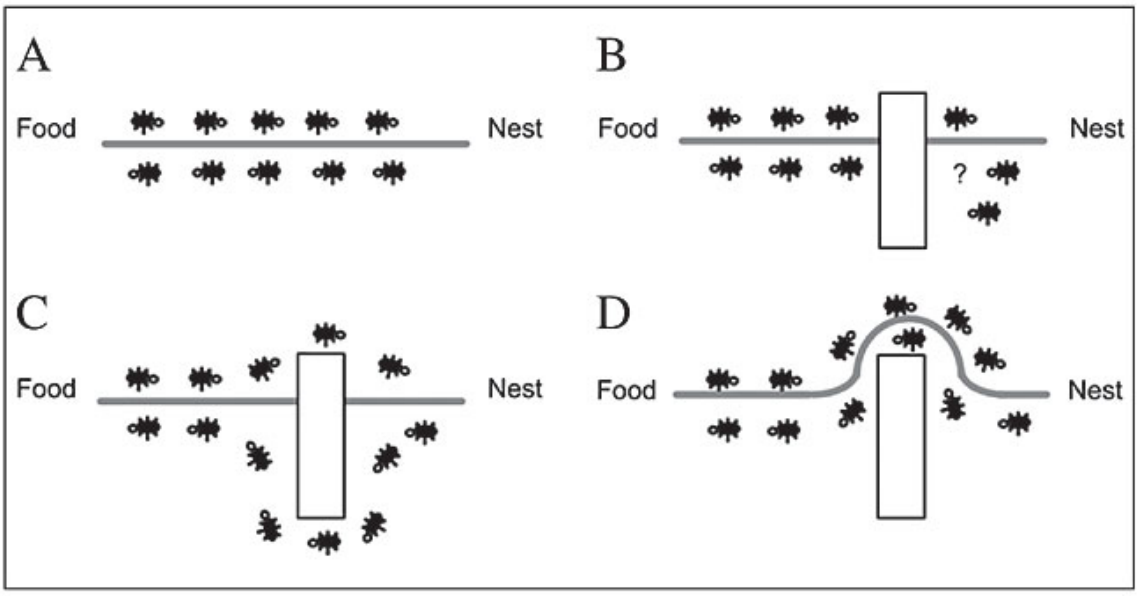

Figure 2. A. Ants in a pheromone trail between nest and food; **B.** an obstacle interrupts the trail; **C.** ants find two paths to go around the obstacle; **D.** a new pheromone trail is formed along the shorter path.

Figure 2.1: Ant Colony

Genetic Algorithms : Genetic Algorithms are general-purpose, robust optimization algorithms that were invented by Holland in the early 1970s; the Darwinian laws of genetics, natural selection, and evolution are used as their operational principles.[23]

In a genetic algorithm, an initial population is chosen at random, with each individual of the population being referred to as a chromosome, these chromosomes represent the solution for the problem; a chromosome is the string of codes where each bit is called as gene and holds information of the problem in coded form, once the population is chosen, each chromosome's fitness is calculated, and the fittest ones are transferred to the next generation while the others are eliminated.[24]

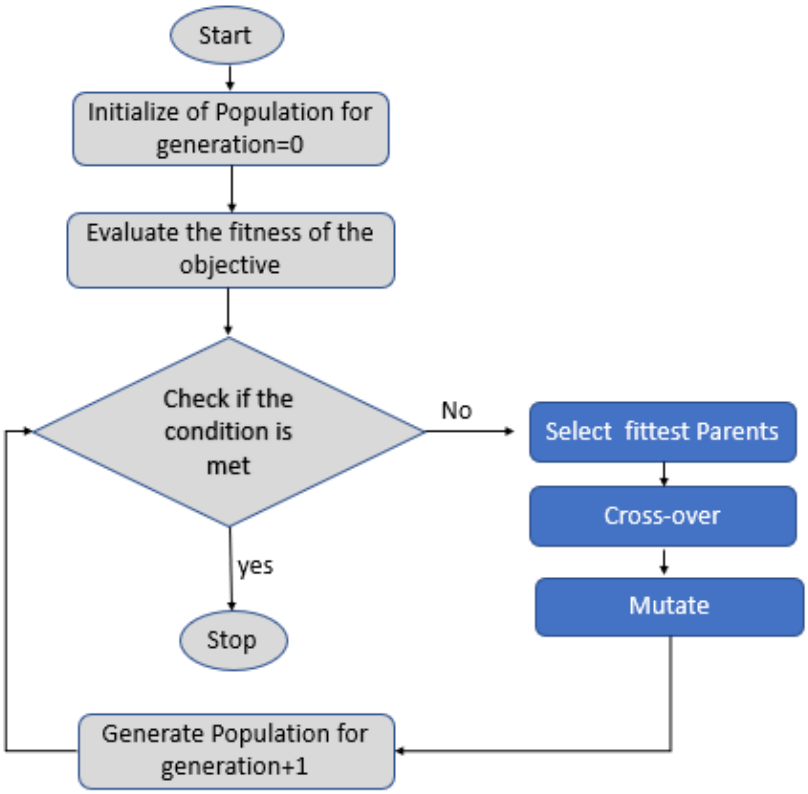

Figure 2.2: Genetic Algorithm

Based on their behavior, the metaheuristic algorithms can be divided into four categories: evolution-based, swarm intelligence based, physics-based, and human-related algorithms.

The principles of natural evolution are imitated by evolutionary algorithms like genetic algorithms and differential evolution to create effective optimization methods. Particle swarm optimization and whale optimization are two examples of swarm algorithms that mimic the collective behavior of various beings. Physical-based algorithms are inspired by real-world physical processes like simulated annealing or gravitational search algorithms, but human behavior-related algorithms are purely inspired by human behavior, such as the League Championship algorithm and learning-based optimization (TLBO) .[21]

2.3 Hybridization

2.3.1 Hybridization Definition

Hybridization is a general model of two or more algorithms that takes advantage of their strengths while minimizing their weaknesses; the combination of techniques does well for tackling a specified problem, given that the obtained results can be improved by these combined techniques on their own. With the hybridization of algorithms, an algorithm's exploitation and exploration can be completely enhanced, for example, an algorithm can cater for the lack of its preciseness and refine the results through synergy with a local search method. [17]

The hybrid approach is becoming more and more common in the field of optimization, and it uses the intention of hybridizing the components from top optimization techniques to enhance the performance of traditional optimization algorithms.

Several factors can be used to categorize hybrid approaches, such as whether the components come from different search paradigms (often constructive or exact methods like constraint programming or integer linear programming) or whether they are homogeneous (e.g., local search or evolutionary methods); a common source of novelty in metaheuristic research is the presentation of a specific hybridization of two or more metaheuristics.[17]

2.3.2 Hybridization Classification

At the first level, we may distinguish between low-level and high-level hybridizations. The functional composition of a single optimization method is addressed via low-level hybridization; in this hybrid class a certain metaheuristic function is changed by another metaheuristic, and the different metaheuristics used in high-level hybrid algorithms are self-contained; the internal workings of a metaheuristic are not directly related.[25]

In relay hybridization, several metaheuristics are used sequentially(one after one), with each taking the output of the previous as its input and functioning as a pipeline; teamwork hybridization represents cooperative optimization models, in which a large number of cooperating agents develop concurrently, each agent searching for a solution space.[25]

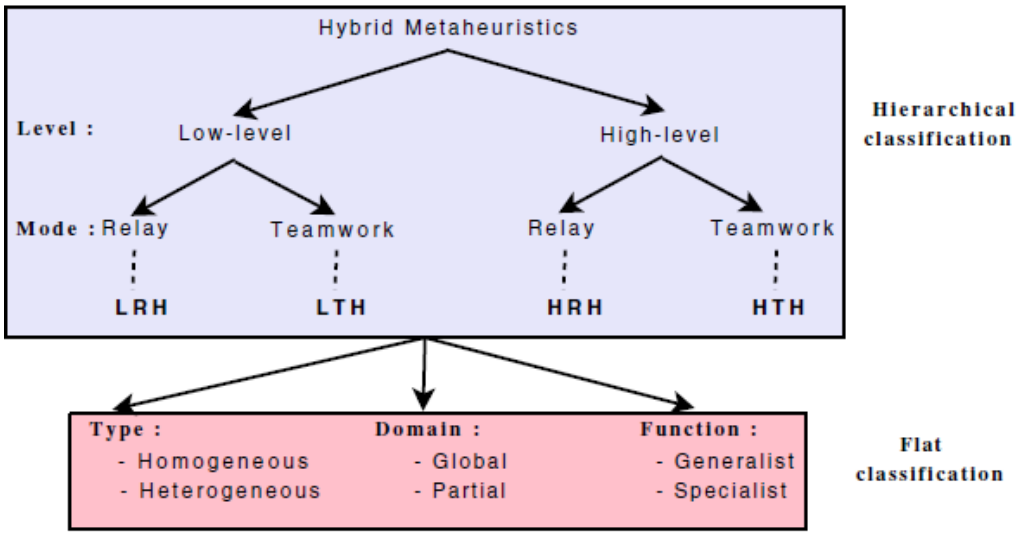

Figure 2.3: Classification of hybrid metaheuristics in terms of design issues

Hierarchical Classification: Four classes are derived from this hierarchical taxonomy:

LRH (Low-level Relay Hybrid): This category of hybrids represents algorithms in which a given metaheuristic is embedded into an S-metaheuristic (Single solution based metaheuristic), like embedding local search into simulated annealing.[25]

LTH (Low-level Teamwork Hybrid): Two competing goals govern the design of a metaheuristic: exploration and exploitation. To provide a reliable estimate of the global optimum, exploration is required to make sure that every area of the space has been fully searched. Exploitation is essential since the refinement of the current solution will often provide a better solution. Evolutionary algorithms, scatter searches, particle swarms, and ant colonies are a few examples of P-metaheuristics (Population-based metaheuristics), which are powerful at exploring the search space and weak in the exploitation of the solutions found.[25]

Therefore, most effective P-metaheuristics have been combined with S-metaheuristics such as local search, simulated annealing, and tabu search, are powerful optimization methods in terms of exploitation. The

strengths and weaknesses of the two categories of algorithms are complementary. The P-metaheuristics will attempt global optimization, whereas the S-metaheuristics will attempt local optimization. A metaheuristic is embedded into a P-metaheuristic in the LTH hybrid. [25]

HRH (High-level Relay Hybrid): Self-contained metaheuristics are executed sequentially in HRH hybrids. For example, another optimization algorithm might produce the initial solution of a given S-metaheuristic. Indeed, the initial solution in S-metaheuristics has a great impact on their performances. Generating the initial solution using greedy heuristics, which often have lower computational costs than iterative heuristics, is a well-known combination scheme.[25]

This hybrid scheme is applied to P-metaheuristics, but to generate a diverse population, a randomized greedy heuristic must be utilized. Greedy heuristics are typically deterministic algorithms, and they always produce the same solution. This has a great impact on P-metaheuristic performance and is carried out explicitly in the scatter search metaheuristic.[25]

P-metaheuristics and S-metaheuristics are frequently combined in the HRH scheme. P-metaheuristics are effective for fast discovering high performance regions in vast and complex search spaces, but they are not ideal for fine-tuning structures close to optimal solutions. S-metaheuristics can then be applied to the high performance structures evolved by the P-metaheuristic.[25]

HTH (High-level Teamwork Hybrid): The HTH scheme covers several self-contained algorithms searching in parallel and cooperating to reach an optimum. It makes sense that HTH will ultimately perform at least as well as one algorithm working alone, if not more often, as one algorithm will be able to help others by exchanging information with them.[25]

Flat Classification: Three classes are derived from the flat classification based on their type, domain, and function :

Homogeneous/Heterogeneous: All combined algorithms in homogeneous hybrids apply the same metaheuristic. Hybrid algorithms such

as the island model for GAs belong to this class of hybrids. The initialization of the homogeneous metaheuristics may be different from their:[25]

Parameters: In general, different parameters are used for the algorithms. For example, in the HTH hybrid scheme which is based on tabu search, the algorithms may be initialized with different tabu list sizes; different crossover and mutation probabilities may be used in evolutionary algorithms, etc

Search components: Given a metaheuristic, one can use different strategies for any search component of the metaheuristic, such as the representation of solutions, objective function approximations, initial solutions, search operators (neighborhood, mutation, crossover, ...), termination criteria, etc.

The robustness of the hybrid algorithm can be improved by using several parameters or search components into a particular metaheuristic.

Global/Partial: we can also distinguish two kinds of cooperation: global and partial. In global hybrids , all the algorithms explore the same whole search space. The goal is here to explore the space more thoroughly, in the sense that all the algorithms solve the whole optimization problem. In partial hybrids, the problem to be solved is decomposed a priory into subproblems, each one having its own search space. Then, each algorithm is dedicated to the search in one of these sub-spaces. Generally speaking, the subproblems are all linked with each other, thus involving constraints between optima found by each algorithm. Hence, the algorithms communicate to respect these constraints and build a globally viable solution to the problem.[25]

Generalist/Specialist: If the algorithms involved all work to solve the same problem, we will speak of a general approach, but if they are launched on different problems, hybridization is then specialized.[25]

2.4 Parallelization

2.4.1 Parallel Architecture

In the last years, parallel computing architectures systems experienced an evolution. This parallelism is achieved by architectures based on shared or based on distributed memory; distributed memory was the first architecture developed and consisted of the use of clusters of a single central processing unit (CPU) computers communicating through a network .[26]

At the beginning of this millennium, shared memory systems have been introduced and are now widely used. Shared memory architectures consist of multiple CPU cores on the same integrated circuit having access to the same global memory. According to the number of cores, these architectures may be classified as Multi-core processors in the case of a lower number of cores processors (two, four, eight, twelve, sixteen...) or Many-core processors, in the case of a larger number of cores. Xeon Phi is considered an example of a many-core processor and may have up to 72 cores, those systems are here referred to as "Multi-core CPU" and "Many-Core CPU" respectively.[26]

One particular case of a many-core CPU is the Graphics Processing Unit (GPU). GPU hardware has a particular architecture and memory management. For that reason, it will be separated into another category and here referred to as "GPU". Programming parallel computing systems, especially heterogeneous ones, is more difficult than sequential programming processors because it depended on the number of cores and communication technologies. To take advantage of parallel architectures, algorithms must be adapted and redesigned to allow task and data parallelism.[26]

2.4.2 GPU Computing

GPUs have achieved great results in recent years, this current hardware was first built to assist video games and 3D graphic applications, but subsequently it has been used for general computational tasks. It is now used in various fields such as data compression, image processing, data mining, etc. The availability of application programming interfaces(API) has made it easier to create parallel applications, but there are still some important principles to follow in order to get efficient performance from GPUs. Understanding the GPU's fundamental parallel programming mechanism is required for designing a parallel application (metaheuristics in our case).[16]

The kernel is the essential unit of a parallel application by which the execution is conducted. It is a piece of code that is called from the CPU (also known as the host) and duplicated on the GPU (also known as the device). The kernel runs within a grid (a collection of blocks), with each block containing a collection of threads.[16]

In NVIDIA's architecture, memory is divided into six types: global memory, constant memory, read-only cache, L2 cache, shared memory, and local memory (registers). The largest memory is global memory, but its frequent use has a negative impact on performance. Constant memory is a global memory with a particular cache that allows a single memory address to be broadcast to all threads in a warp. A read-only cache is a type of data cache that allows users to access read-only data in global memory with lower latency. The L2 cache handles streaming multiprocessor load, store, atomic, and texture instruction requests. Shared memory is a 64-KB on-chip memory with extremely low latency. Local memory(registers) is the fastest but also the smallest type of memory. Each thread has its local memory that is not accessible to other threads.[16]

2.4.3 Message Passing Interface(MPI)

A message passing interface(MPI) is a common software model for parallel environments. MPI is a message passing application developer interface that includes semantic judgments and a protocol for how its highlights must be carried out in any implementation, MPI plans to maintain scalability, performance, and portability.[27]

The Message Passing Interface(MPI) standard was developed by the Message Passing Interface forum. To develop parallel computing in a distributed memory environment, MPI is the appropriate tool to utilize because it has the required standard libraries. MPI's first version was launched in 1994, followed by MPI-2 in 1997, MPI is capable of both collective and point-to-point communication. In the parallel programming environment, MPI Library has hundreds of function interfaces executed by computers such as different clusters interacting with one another, it is supported by multiple programming languages, including FORTRAN, C, and C++, which can call the MPI library directly. Scalability and high performance are the main goals of message passing interfaces. The following is the message forwarding interface principles: point-to-point communication, process groups, collective operations, process topologies, communication context, environmental management, and inquiry and profiling interface.[27]

2.4.4 Parallel Metaheuristic

The performance of most metaheuristics is not scalable; when dealing with high dimensional problems, it suffers in terms of both time complexity and effectiveness; to overcome this limit, GPU-based parallel metaheuristics have been proposed. The scientific community is becoming more and more interested in this latter technique to reduce the execution time and to improve the quality of the solutions found[16]. There are three categories for the parallel design of metaheuristics:

Algorithmic level: This level enables the parallel execution of many algorithms; the algorithms are capable of independently running with various starting solutions and/or parameters and selecting the best run results. In this situation, we speed up the execution time and achieve the same result as if we had performed all of these algorithms sequentially. The behavior of the metaheuristics might vary as a result of the algorithms' ability to cooperate, which enhances the expected quality of the resulting solutions.[16]

Iteration Level: At this level, parallelization is allowed during each iteration; this parallelizes the neighborhood generation and/or evaluation processes, and different areas of the neighborhood are executed in parallel. The metaheuristic continues to operate in the same manner. The primary objective is to accelerate the algorithm by reducing the search time.[16]

Solution Level: This level enables the parallelization of a single solution; for example, to evaluate the constraints or objective function for a generated solution, the metaheuristic continues to operate in the same manner. The basic goal is to accelerate the search.[16]

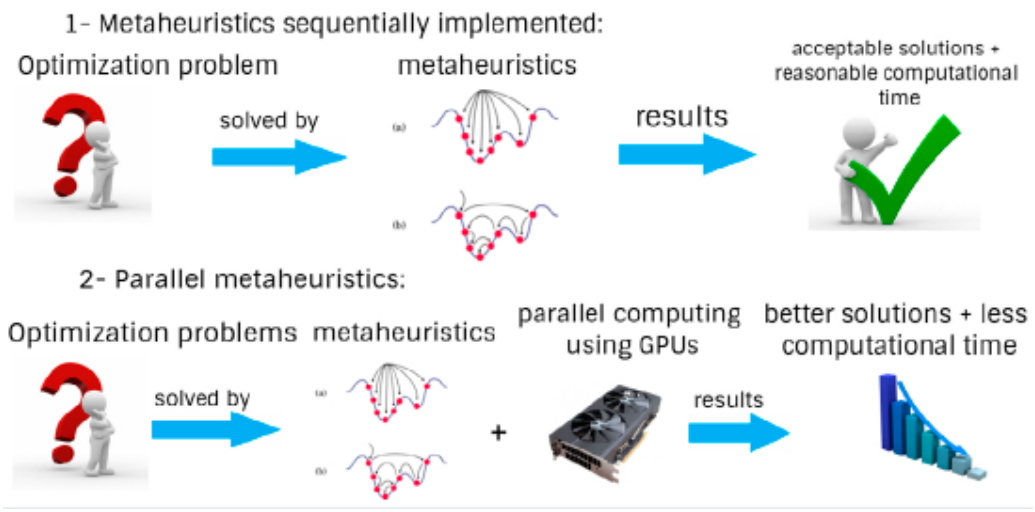

Figure 2.4: GPU parallelization strategies for metaheuristics

2.4.5 Parallel Metaheuristic Classification

When multiple processes are working simultaneously on multiple processors to handle a single instance of a problem and try to find the best (or a) solution for it, this is referred to as parallel, distributed, or concurrent computing. Parallelism results from the decomposition of the overall computing load and the assignment of the resulting tasks to available processors. The term fine or coarse-grained parallelization refers to how "small" or "large" the tasks are in terms of algorithm work or search space.[28]

The algorithm, the search space, or the problem structure may be the subject of the decomposition.

Functional parallelism: Corresponds to the first case, where the computing intensive parts of the algorithm are divided into several tasks(processes), working on the same data or particular parts of the data, are assigned to separate processors and execute in parallel, perhaps exchanging information. The primary source of functional parallelism for metaheuristics is the concurrent execution of the innermost loop iterations, such as evaluating neighbors, computing an individual's fitness, or having ants forage concurrently.[28]

This is frequently the only source of readily available parallelism in metaheuristics, as the execution of most other steps in the algorithm depends on the status of the search, what has been done up to this point, and the values of the decision variables, which necessitates either the computation of the previous steps to be completed, or the synchronization of computations; synchronization typically yields significant delays, which may make such parallel computation irrelevant.[28]

Functional parallelism has historically been appealing as a low-level component of hierarchical parallelization strategies or when addressing problem settings that need a significant part of the computational work to be spent in inner-loop algorithmic components. The rapid growth of graphical processing units (GPU), which are now prevalent in most computers, is modifying this assertion as very substantial reductions in computing times can be achieved.[28]

Search space separation: A second important class of parallel strategies. The search space and the problem structure are two other situations which belong within this category. The fundamental concept is to divide the problem domain or associated search space and then address the problem on each of the resulting components using a specific solution methodology. In fact, there are no data dependencies between the evaluation functions of various solutions, allowing for parallel computation of these functions.[28]

Furthermore, when a processor is assigned to each solution, theoretically, the parallelism in the solution or search space is as large as the space itself. The latter strategy is obviously impractical, thus the search space is divided into subspaces and assigned to several different processors. An exact or heuristic search method is needed to implicitly explore the search space left by such a separation because it is too large for explicit enumeration for each processor.[28]

2.5 Iterated Local Search

2.5.1 Iterated Local Search Definition

Iterated Local Search(ILS) is a well-known single-solution based metaheuristic that works well for combinatorial optimization problems(COPs), such as issues with logistics, transportation, scheduling, health care, and marketing. It is an effective method

because it has many desirable characteristics, such as accuracy, speed, simplicity, and flexibility.[29]

The Local Search(LS) algorithm, which includes identifying an initial solution and performing a neighborhood search until a local optimal solution is discovered, is extended by the Iterated Local Search algorithm; each individual problem requires the definition of a neighborhood, which is a set of solutions that can be produced from the existing solution by performing slight modification. It is rather easy to find an initial solution, define a neighborhood, and define an LS for all or almost all COPs.[29]

The fundamental problem with such a straightforward LS method is that the quality of the solutions is typically poor and the local optimal solution is frequently distant from the global optimal solution. ILS is a multi-start metaheuristic that, by combining more sophisticated methods, improves the basic random restart, by using a walk that moves from one local optimal solution to a nearby one, it aims to overcome the disadvantages of random restart. To implement this idea, another phase is included in the LS that allows to "restart" the search but not to "lose" the good properties and components of the solutions already obtained.[29]

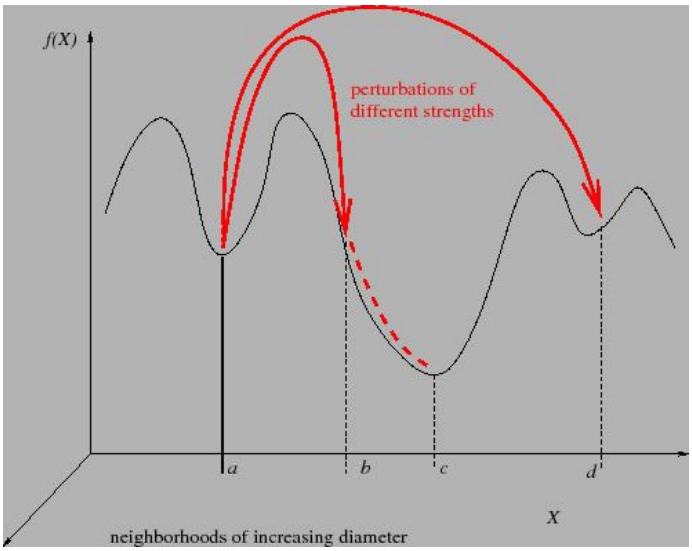

Figure 2.5: Iterated Local Search

Implementation Details

s(0) = GenerateInitialSolution
$s* = LocalSearch(s_0)$
REPEAT
s' = Perturbation(s*,history)
s*' = LocalSearch(s')
s* = AcceptanceCriterion(s*,s*',history)
UNTIL termination condition met

Figure 2.6: ILS Psuedo Code

The ILS algorithm is composed of four main components: GenerateInitialSolution, LocalSearch, Perturbation, and AcceptanceCriterion. A simple implementation of the ILS for a COP can be quite straightforward to design. The four main components can be defined as follows:

Generate Initial Solution: Because random solutions are often of low quality, it is necessary, to begin with the best possible solution. A greedy heuristic is a suitable choice for this since it typically produces solutions of higher quality than random solutions. To obtain the initial solution, additional metaheuristics or approximation methods can be applied. While this does not guarantee the quality of the final local optimal solution, it can reduce the search process.[30]

Local Search: The local research method chosen has a significant impact on the overall performance of an ILS. An important consideration while constructing an ILS is the equilibrium between the local search phase and the number of iterations. A faster and more frequent local search algorithm may be more effective than a slower and more powerful one. As a result, the Local Search phase's design and interaction with the other ILS components need to be carefully done and properly considered, and studied.[29]

Perturbation: The main goal of the Perturbation phase is to escape from the local optimal solution area by applying perturbations or changes in the current local optimal solution. It is important to ensure that the obtained solution is not undoable by the local search and should complement it in some way. The design of the Perturbation phase is one of the most important issues when

implementing an ILS algorithm, as small perturbations can lead to large computational times and a random restart type algorithm, and large perturbations can lead to jumping from one solution to another without descent to a good local optimal solution.[30]

AcceptanceCriterion: AcceptanceCriterion is a procedure that determines if a solution is accepted or not. It controls the balance between intensification and diversification of the search. Two strategies are possible: improvement type descent which only accepts better solutions or the opposite extreme of accepting the new solution regardless of its cost. There are many intermediate choices between these two extreme cases are possible, and in particular rather complex acceptance criteria that involve a limited amount of directed diversification or intensification are also possible.[29]

2.6 Accelerated Particle Swarm Optimization

2.6.1 APSO Definition

To update particle velocity, the standard PSO uses both the personal best position pi and the current global best x*. The objective of using the personal best position is to improve the swarm's diversity; however, this diversity can be achieved with some randomness. As a result, there is no compelling reason to use the personal best position. Yang suggested a simplified version of PSO known as accelerated particle swarm optimization (APSO) in 2008 to accelerate the algorithm's convergence, in which only the global best solution is involved[31]. In APSO, velocity is given by:

$$v_i{}^t + 1 = v_i{}^t + b(x * -x_i{}^t) + a(e - (1/2) \tag{2.1}$$

The APSO can be further improved by formulating the update of particle's position in a single step as mentioned in:

$$x_i{}^t + 1 = x_i{}^t + b(x * -x_i{}^t) + a(e - (1/2) \tag{2.2}$$

Equation (2.2) consists of three terms. The first term determines the current position of the ith particle. The second term refers to a social component of moving the particle i toward the position of the current global best particle, while the third term is connected with the randomized move of the ith particle within the search

space. In the second term of (2.2), the parameter b expresses the attractiveness of the global best solution. Its value can be increased gradually from 0 to 1 to speed up the convergence of APSO.[32] In the third term of(2.2), a is a randomization parameter and e is a random number generator uniformly distributed in the range [0, 1].

2.6.2 APSO Algorithm

The algorithm begins by setting the initial values for the attraction b and randomization a parameters. The initial population of N particles (i = 1, 2,..., N) is then produced at random, with each solution starting at xi. The fitness function value of each solution is used to evaluate the initial swarm. The best solution x* is then initialized by all of the particles' best solutions. The APSO moves on from the initial set of solutions to the optimal solution iteratively by updating the position xi of each particle i. Based on the continuous flying iterations, solutions are continuously changed.[32]

This is repeated till the termination criteria are satisfied. Finally, when all criteria are successfully met the best so far particle x is reported.

Fitness function f(x)
Define the attraction parameter b,
Randomization parameter a
Generate an initial population of N particles
while (t¡ Max number of iteration) **do**
for i¡– 1 to N **do**
Calculate new location(2.2)
Evaluate fitness function at new locations
end for
Rank the particles and find the current global best x*
end while

Figure 2.7: Psudo code of APSO algorithm

2.7 Conclusion

In this chapter, we presented the metaheuristics, hybridization, and parallelization which will be used to enhance our approach, we also define the Iterated Local Search and Accelerated Particle Swarm Optimization.

In the next chapter, we will present our application which is the proposed solution for the query expansion problem and discuss the results and evaluate the work.

Chapter 3

Experimental and Results

3.1 Introduction

In this chapter, we detailed our work starting with a simplified explanation of the implementation phase and the used tools: programming languages, and libraries we used, we mentioned also CUDA and MPI4Py, then we define the dataset we used which is CISI.

Then, we moved to the system implementation where we mentioned the evaluation metrics, which will be used to measure the quality of this work, then we talked about our experiment, and how we make this work from the beginning, and after that we will present the obtained results.

3.2 Implementation

3.2.1 Programming Language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms; and can be freely distributed.

Over the last few years, Python has become increasingly popular in the scientific community, which is largely due to the availability of performance-oriented libraries like NumPy, SciPy, TensorFlow, or sci-kit-learn. However, for compute-intensive operations that cannot be accelerated using specialized libraries, pure interpreted Python can be very slow. Moreover, for CPU-bound tasks multi-threading is generally inefficient, because of Python's global interpreter lock (GIL). Therefore, several projects like PyPy, Cython, Numba, and Nuitka aim at increasing Python's performance.

Numba is an alternative implementation of Python using JIT compilation to translate subsets of Python and NumPy to fast machine code. However, instead of re-

placing the Python interpreter, Numba provides decorators that are inserted into the code to trigger the LLVM-based JIT compilation of selected functions.

3.2.2 MPI4Py

MPI for Python provides Python bindings for the Message Passing Interface (MPI) standard, allowing Python applications to exploit multiple processors on workstations, clusters, and supercomputers.

This package builds on the MPI specification and provides an object oriented interface resembling the MPI-2 C++ bindings. It supports point-to-point (sends, receives) and collective (broadcasts, scatters, gathers) communication of any pickable Python object, as well as efficient communication of Python objects exposing the Python buffer interface (e.g. NumPy arrays and builtin bytes/array/memoryview objects).

3.2.3 CUDA Threading Model

CUDA (Compute Unified Device Architecture) is a parallel computing environment, which provides an application programming interface for NVIDIA architectures. The notion of thread in CUDA doesn't have exactly the same meaning as CPU thread. A thread on GPU is an element of the data to be processed. Compared to CPU threads, CUDA threads are lightweight. That means that changing the context between two threads is not a costly operation.[33]

Threads are organized within so called thread blocks. A kernel is executed by multiple equalThe figure thread blocks. Figure below illustrates these multiple blocks organization. Blocks can be organized into a one-dimensional or two-dimensional grid of thread blocks, and threads inside a block are regrouped similarly. First, the advantage of grouping is that the number of blocks processed simultaneously by the GPU is closely linked to hardware resources. Secondly, since each thread is provided with a unique id that can be used to compute different data, this model of threads provides an easy abstraction for SIMD architecture.[33]

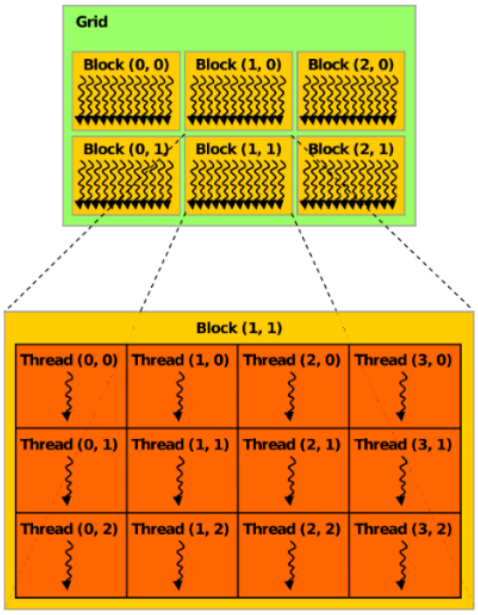

Figure 3.1: CUDA threads model

3.2.4 Dataset

The data were collected by the Centre for Inventions and Scientific Information ("CISI") and consisted of text data from about 1,460 documents and 112 associated queries. Its purpose is to be used to build models of information retrieval where a given query will return a list of document IDs relevant to the query.

3.3 The system implementation steps

3.3.1 Evaluation Metrics

Precision

it's the percentage of documents that are truly related to the query.

$$Precision = \frac{|Relevantdocuments \cap Retrieveddocuments|}{|Retrieveddocuments|} \tag{3.1}$$

Recall

It is the percentage of documents that have been found and are relevant to the query.

$$Precision = \frac{|Relevantdocuments \cap Retrieveddocuments|}{|Relevantdocuments|} \tag{3.2}$$

Precision@k

Precision usually takes into account all of the documents that have been retrieved. However, another method of calculating precision involves a cut-off rank, k. We calculate precision just for the top kdocuments using this method. This is known as precision at k, or P(k). Consider an information retrieval model that accepts a query and returns documents that are similar to that query to better understand P(k).

Mean Average Precision

We use the mean average precision (mAP) to measure the accuracy of information retrieval models. The mAP is a value between 0–1 0–1, with higher scores representing a more accurate model.We describe it by the following formula:

$$mAP = (\frac{1}{N}) \sum_{i=1}^{N} AP_i \tag{3.3}$$

In the above formula, N is the total number of queries, and APi is the average precision of the query. In simple terms, mAP is the average of average precisions across allqueries.

Mean Reciprocal Rank(MRR)

The Mean Reciprocal Rank (MRR) is a relative score that computes the average or mean of the inverse of the ranks at which the first relevant document for a set of queries was retrieved. It is related to the use case in which the user only wants to see one relevant document for search.

3.3.2 Experimental

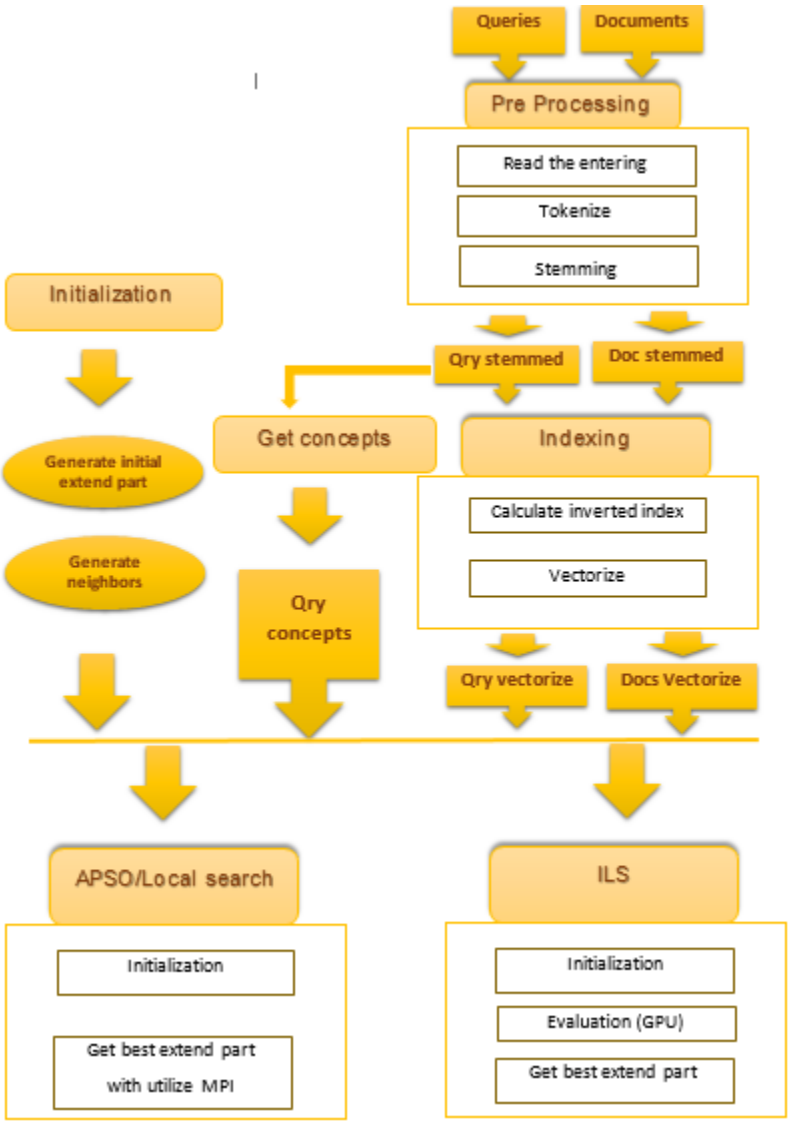

Figure 3.2: Implementation System

Preprocessing

In the preprocessing step, first, we read the entered data which are queries and documents; and extract the words from both of them, after that, we tokenize the extracted words because they come as a sequence of symbols or characters, so we use a special NLP tool called "Tokenizer" to extract words.

In this step, we used a tokenizer to convert text to lowercase, split it into word entries in both documents and queries as word lists and we removes stop words as well.

the last step of preprocessing is called stemming, it takes word matching one step further and tries to map related words and this means not just the forms of the very same word.

As a result, we have as sorts: query stemmed list and document stemmed list.

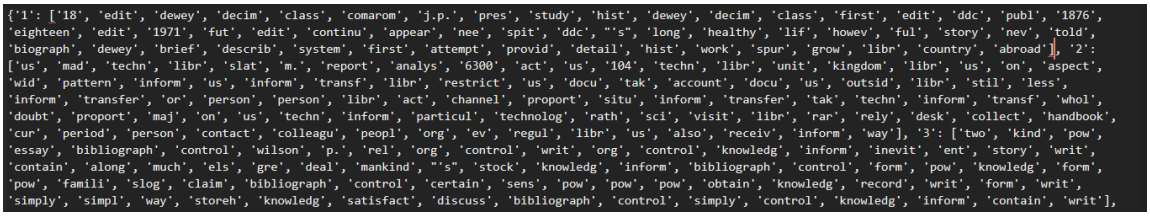

Figure 3.3: Document stemmed list

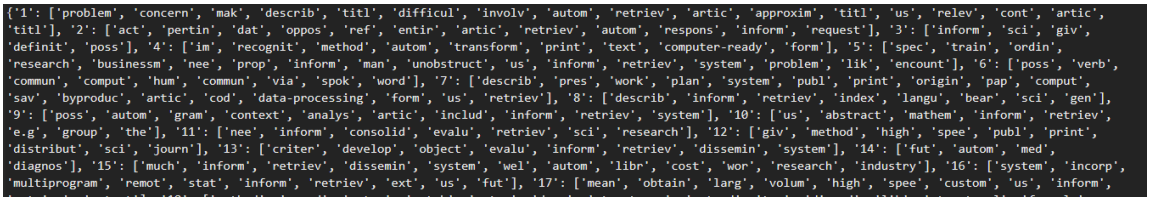

Figure 3.4: Query stemmed list

Indexing

Term frequency(tf) is the occurrence of the term in a document, the more frequently the term is used in a document, the more relevant this document becomes

to the query. Not all terms are equally important, even after stopwords are removed there are still terms that are frequently used across all documents.

Inverse document frequency(idf) is the frequency of occurrence of the term across all documents in the collection; the higher the number of documents that contain the term, the lower its discriminative power.


```
{"''": 0.7595191391644989, "'60": 2.8633228601204554, "'70": 2.8633228601204554, "'anyhow": 2.8633228601204554, "'apparent": 2.8633228601204554, "'basic":
2.6872316010647745, "'better": 2.8633228601204554, "'bibliograph": 2.8633228601204554, "'bibliometrics": 2.8633228601204554, "'building": 2.8633228601204554,
"'built-in": 2.8633228601204554, "'citation": 2.8633228601204554, "'citeable": 2.8633228601204554, "'community": 2.8633228601204554, "'comparative":
2.8633228601204554, "'compare": 2.8633228601204554, "'completeness": 2.8633228601204554, "'computer-based": 2.8633228601204554, "'computers": 2.8633228601204554,
"'consensual": 3.1643528557844367, "'cosmopolitan": 2.8633228601204554, "'cramming": 2.8633228601204554, "'crisis-state": 2.8633228601204554, "'criteria":
2.8633228601204554, "'describing": 2.8633228601204554, "'document": 2.8633228601204554, "'documentalistics": 2.8633228601204554, "'documentation":
2.8633228601204554, "'failure": 2.8633228601204554, "'force": 2.8633228601204554, "'general-dynamic": 2.8633228601204554, "'half": 2.8633228601204554,
"'half-life": 2.465382851448418, "'humanities": 2.8633228601204554, "'hyperbolic": 2.8633228601204554, "'implicit": 2.8633228601204554, "'in":
2.8633228601204554, "'indirect": 2.8633228601204554, "'inevitable": 2.8633228601204554, "'informatics": 2.8633228601204554, "'information": 2.8633228601204554,
"'intellectual": 2.8633228601204554, "'internal": 2.8633228601204554, "'known-item": 2.8633228601204554, "'language": 2.8633228601204554, "'law":
2.6872316010647745, "'level": 2.8633228601204554, "'library": 2.6872316010647745, "'local": 2.8633228601204554, "'locating": 2.8633228601204554, "'maps":
2.8633228601204554, "'marcas": 2.8633228601204554, "'marshalling": 2.8633228601204554, "'meaning": 2.8633228601204554, "'normal": 2.8633228601204554, "'nuclear":
2.8633228601204554, "'objective": 2.8633228601204554, "'obsolescence": 2.8633228601204554, "'performance": 2.8633228601204554, "'practicum": 2.8633228601204554,
"'pure": 2.8633228601204554, "'reader": 2.8633228601204554, "'relevance": 2.8633228601204554, "'retrieval": 2.8633228601204554, "'s": 0.701954857885481,
```


Figure 3.5: Document ifd

The second step of indexing is vectorize, it's used to apply idf weighting to the document terms.


```
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8323983947288827, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.1643528557844367, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.6510075300943634, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 2.5388685624650265, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
```


Figure 3.6: Document vectorizing

{'1': [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.701954857885481, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 2.2101103463451124, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.6872316010647745, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.8633228601204557, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,

Figure 3.7: Query vectorizing

Get Concept

In this step, we extract the concepts which they are query neighbors from ConceptNet, then we store them in a dictionary having the query id and list of these concepts.

best extend part: [['library_work', 'natural_languag', 'finding_inform', 'legal', 'adult', 'concept', 'detect', 'tactic', 'take_examin', 'organizing_th']]

Figure 3.8: Best Extended Part

Generated voisinages: [[['na', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'dob'], ['data', 'na', 'archaic',
'programming_languag', 'research_octane_numb', 'inherit', 'collect'], ['data', 'na', 'archaic', 'programming_languag', 'research_octane_numb',
'undertaken_at_mit', 'report'], ['data', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'for_report'], ['na',
'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'dob'], ['data', 'archaic', 'programming_languag',
'research_octane_numb', 'undertaken_at_mit', 'inherit', 'legal'], ['data', 'na', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit',
'examin'], ['data', 'na', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'library_work'], ['data', 'na', 'archaic',
'programming_languag', 'undertaken_at_mit', 'inherit', 'find'], ['data', 'na', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit',
'for_report'], ['data', 'na', 'archaic', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'lab'], ['data', 'na', 'archaic', 'research_octane_numb',
'undertaken_at_mit', 'inherit', 'advance_knowledg'], ['na', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'look'],
['na', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'information_gath'], ['data', 'na', 'archaic',
'research_octane_numb', 'undertaken_at_mit', 'inherit', 'scrutini'], ['data', 'na', 'archaic', 'programming_languag', 'undertaken_at_mit', 'inherit',
'visit_museum'], ['data', 'na', 'archaic', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'knowledg'], ['data', 'na', 'programming_languag',
'research_octane_numb', 'undertaken_at_mit', 'inherit', 'inquiri'], ['data', 'na', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit',
'for_studi'], ['na', 'archaic', 'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'affect'], ['na', 'archaic',
'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'data_collect'], ['data', 'na', 'archaic', 'programming_languag',
'undertaken_at_mit', 'inherit', 'inspir'], ['data', 'na', 'archaic', 'programming_languag', 'undertaken_at_mit', 'inherit', 'no'], ['data', 'na',
'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'report'], ['data', 'na', 'archaic', 'programming_languag',
'research_octane_numb', 'undertaken_at_mit', 'research'], ['data', 'na', 'archaic', 'programming_languag', 'research_octane_numb', 'inherit', 'inform'], ['data',
'na', 'archaic', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'natural_languag'], ['data', 'na', 'archaic', 'programming_languag',
'research_octane_numb', 'inherit', 'chang'], ['data', 'na', 'archaic', 'programming_languag', 'undertaken_at_mit', 'inherit', 'out'], ['na', 'archaic',
'programming_languag', 'research_octane_numb', 'undertaken_at_mit', 'inherit', 'chang']]]

Figure 3.9: Generate neighbors

Initialization

In this step, we used two functions; the first one will generate the initial extended part. and the other will generate the initial extended part's neighbors and we store

them in a dictionary. We call neighbor every extended part that differs from the initial extended part by a certain number of n-words.

Figure 3.10: Neighberhood Dictionary

Parallel ILS Using GPU

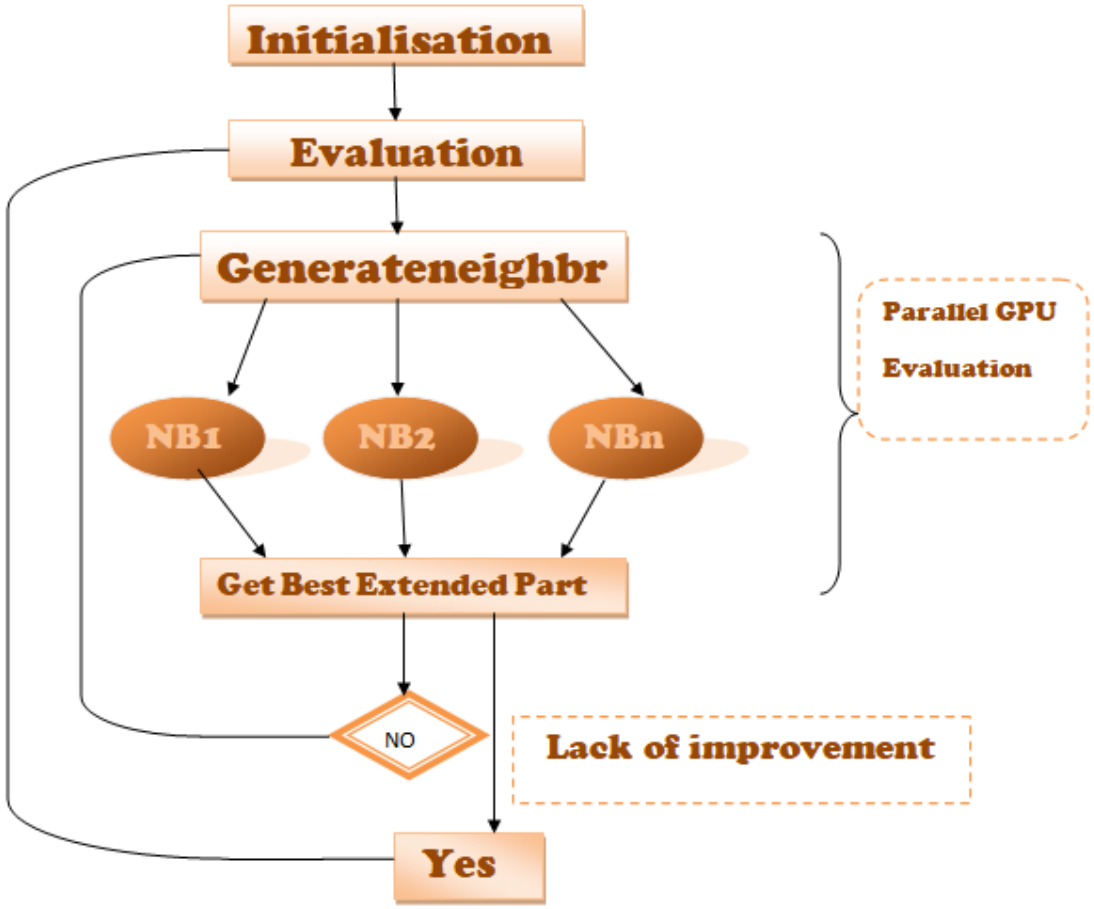

Figure 3.11: ILS-GPU

We have implemented a parallel low-level algorithm ILS using GPU in order to reduce the execution time of the evaluation process, in this step, we started with the initialization of a random extended part, then we update it with GPU; we get the best extended part by evaluation then we generate his neighbors and keep repeating the same process until we get the best extended part, then we do a perturbation for exploring more search areas, we keep searching as long as the new gmax is greater than old gmax; if the best extended query lacks improvement we will re-evaluate it and redo the process until we get the best extended part that satisfies the user's need.

Parallel APSO Using MPI

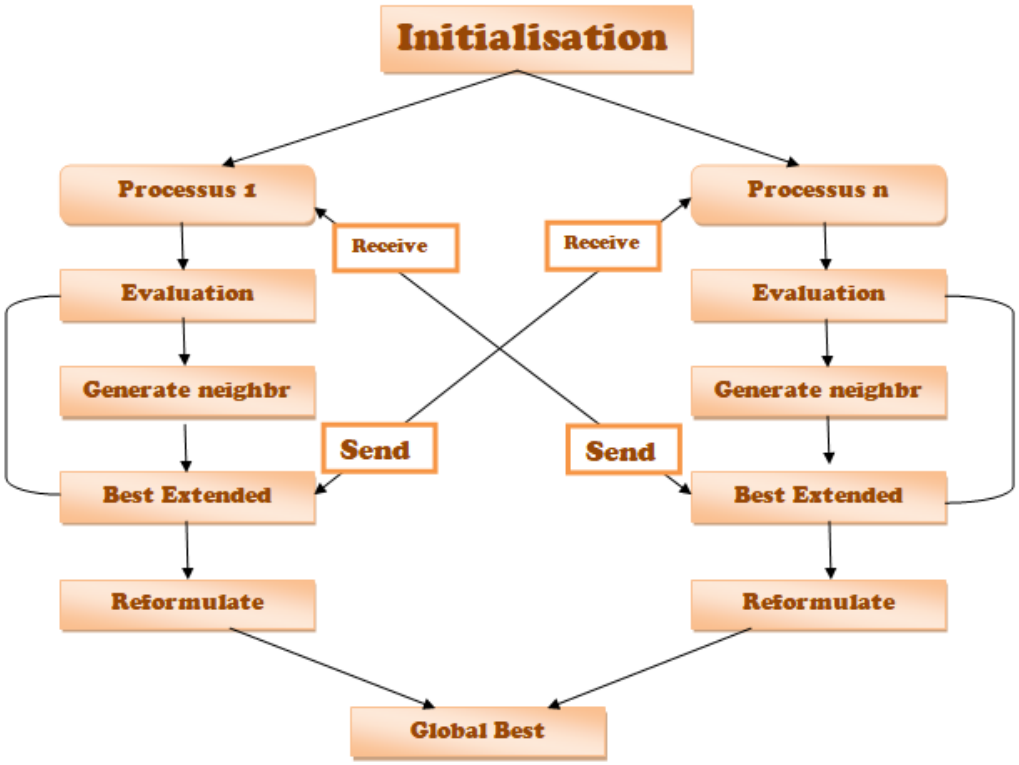

Figure 3.12: APSO with Local Search -MPI

We have implemented a parallel high-level algorithm APSO with a local search using MPI, we wanted to launch multiple processus here but due to the lack of material, we could just launch two processus. In this step, we send one solution to each processus, and every solution will generate its neighbors to get the best local solution when he gets it, he will send it to the other processus, and the two processus will update their solution according to the global solution and their local solution, after that, they will generate new neighbors for the new solutions and do a local search when they find another local solution they will keep doing the same process until the number of iterations ends or all the obtained solution are equal. The final

process here is to compare their results and choose the best one of them as a "Global Best".

3.3.3 Results

	Original Query	ILS-GPU	APSO-LS-MPI
MAP	0.10785475676540142	0.12768650921895297	0.1259851132750418876
MRR	0.43312049062049063	0.5607434640522875	0.46522444946357994

Table 3.1: Test Results

We note that there is a clear improvement between the original using Baseline Model, iterated local search with GPU, and accelerated particle swarm with a local search using MPI; although the resources used are somewhat scarce.

For APSO, we could not use more than two processors, but MPA got the best value. This demonstrates the effectiveness of the method of using the parallel accelerated particle swarm optimization with serial local search.

For GPU, we relied on a certain number of neighbourhoods. Due to time constraints, we were not allowed to determine the optimal values of the neighbourhood and number of the blocks and the threads in every block.

	Original Query	ILS-GPU	APSO-LS-MPI
P@5	0.8	0.6	0.8
P@20	0.45	0.4	0.5

Table 3.2: p@k comparaison

	CPU	GPU[10,10]	GPU[20,20]	GPU[30,30]	GPU[40,40]
time	2139.477	258.588	260.359	264.574	142.581

Table 3.3: Time compraison

In this table we used 50 neighborhoods and 50 iterations, where we noticed that there is a wide difference between the speed of implementation of CPU and GPU, this is evidence that the use of parallelism is very useful in the process of evaluation, as well as the increase in the number of the blocks and threads within each block increases the effectiveness of the execution time.

3.4 Conclusion

In this chapter, we talked about how we built our approach based on the Iterated Local Search algorithm and Accelerated Particle Swarm Optimization with Local Search.

The obtained results were satisfying and led to many possible future improvements at several levels, such as the iteration number, the neighborhood generation, and the selection of the initial extended part.

General Conclusion

We achieved our goal of using ConceptNet to develop parallel applications based on query expansion in this study. After studying and researching, we implemented a parallel Iterated Local Search (ILS) with GPU and a parallel Accelerated particle swarm optimization (APSO) with Local Search (LS) using MPI. We tested our approach on the CISI dataset, which proved that our approach was good and not expensive, but we noticed that its effectiveness was limited due to a lack of materials.

We concluded that programming and implementing parallel machines differ from implementing standard machines in that the characteristics of the machine and the targeted programs, as well as the ability to merge programming technologies, make this project larger than implementing a standard program.

We want in the future to test this program on good machines that will allow us to measure the effectiveness of these programs, we will be able to reset the parameters of our program such as the length of the extended part, number of neighbors, number of iterations, etc and we hope that we apply this program to other datasets such as TREC-3, CACM, FIREBASE, etc.

We planned to construct a hybrid algorithm combining ILS and APSO with the two MPI and GPU technologies, but due to the lack of time and techniques we couldn't achieve this goal.

Bibliography

[1] A. Roshdi and A. Roohparvar, "Review: Information Retrieval Techniques and Applications," *International Journal of Computer Networks and Communications Security*.

[2] S. Ibrihich, A. Oussous, O. Ibrihich, and M. Esghir, "A Review on recent research in information retrieval," *Procedia Computer Science*, vol. 201, pp. 777–782, 2022.

[3] H. Kaur and V. Gupta, "Indexing process insight and evaluation," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, (Coimbatore, India), pp. 1–5, IEEE, Aug. 2016.

[4] R. Agarwal, K. Arya, S. Shekhar, and R. Kumar, "An Efficient Weighted Algorithm for Web Information Retrieval System," in *2011 International Conference on Computational Intelligence and Communication Networks*, (Gwalior, India), pp. 126–131, IEEE, Oct. 2011.

[5] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni, "An Introduction to Information Retrieval," in *Web Information Retrieval*, pp. 3–11, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[6] B. Saini, V. Singh, and S. Kumar, "Information Retrieval Models and Searching Methodologies: Survey," vol. 1, no. 2.

[7] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: A survey," *Information Processing & Management*, vol. 56, pp. 1698–1735, Sept. 2019.

[8] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma, "Query expansion by mining user logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, pp. 829–839, July 2003.

[9] I. Rasheed, H. Banka, and H. M. Khan, "Pseudo-relevance feedback based query expansion using boosting algorithm," *Artificial Intelligence Review*, vol. 54, pp. 6101–6124, Dec. 2021.

[10] M. A. Raza, R. Mokhtar, N. Ahmad, M. Pasha, and U. Pasha, "A Taxonomy and Survey of Semantic Approaches for Query Expansion," *IEEE Access*, vol. 7, pp. 17823–17833, 2019.

[11] M. A. Khedr, F. A. El-Licy, and A. Salah, "Ontology based Semantic Query Expansion for Searching Queries in Programming Domain," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, 2021.

[12] D. K. Sharma, R. Pamula, and D. S. Chauhan, "Semantic approaches for query expansion," *Evolutionary Intelligence*, vol. 14, pp. 1101–1116, June 2021.

[13] G. Jain and A. Bansal, "Common sense based automatic query expansion," *Journal of Information and Optimization Sciences*, vol. 41, pp. 1579–1587, Oct. 2020.

[14] A. Kotov and C. Zhai, "Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries,"

[15] A. Bouchoucha, J. He, and J.-Y. Nie, "Diversified query expansion using conceptnet," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, (San Francisco, California, USA), pp. 1861–1864, ACM Press, 2013.

[16] M. Essaid, L. Idoumghar, J. Lepagnot, and M. Brévilliers, "GPU parallelization strategies for metaheuristics: a survey," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, pp. 497–522, Sept. 2019.

[17] J. Swan, S. Adriaensen, A. E. Brownlee, K. Hammond, C. G. Johnson, A. Kheiri, F. Krawiec, J. Merelo, L. L. Minku, E. Özcan, G. L. Pappa, P. García-Sánchez, K. Sörensen, S. Voß, M. Wagner, and D. R. White, "Metaheuristics "In the Large"," *European Journal of Operational Research*, vol. 297, pp. 393–406, Mar. 2022.

[18] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, "Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019)," *IEEE Access*, vol. 9, pp. 26766–26791, 2021.

[19] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?," *Swarm and Evolutionary Computation*, vol. 54, p. 100671, May 2020.

[20] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022.

[21] M. Abd Elaziz, A. H. Elsheikh, D. Oliva, L. Abualigah, S. Lu, and A. A. Ewees, "Advanced Metaheuristic Techniques for Mechanical Design Problems: Review," *Archives of Computational Methods in Engineering*, vol. 29, pp. 695–716, Jan. 2022.

[22] Q. Luo, H. Wang, Y. Zheng, and J. He, "Research on path planning of mobile robot based on improved ant colony algorithm," *Neural Computing and Applications*, vol. 32, pp. 1555–1566, Mar. 2020.

[23] G. Papazoglou and P. Biskas, "Review and Comparison of Genetic Algorithm and Particle Swarm Optimization in the Optimal Power Flow Problem," *Energies*, vol. 16, p. 1152, Jan. 2023.

[24] H. Malik, A. Iqbal, P. Joshi, S. Agrawal, and F. I. Bakhsh, eds., *Metaheuristic and Evolutionary Computation: Algorithms and Applications*, vol. 916 of *Studies in Computational Intelligence*. Singapore: Springer Singapore, 2021.

[25] E.-G. Talbi, ed., *Hybrid Metaheuristics*, vol. 434 of *Studies in Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[26] J. Gmys, T. Carneiro, N. Melab, E.-G. Talbi, and D. Tuyttens, "A comparative study of high-productivity high-performance programming languages for parallel metaheuristics," *Swarm and Evolutionary Computation*, vol. 57, p. 100720, Sept. 2020.

[27] T. Ragunthar, P. Ashok, N. Gopinath, and M. Subashini, "A strong reinforcement parallel implementation of k-means algorithm using message passing interface," *Materials Today: Proceedings*, vol. 46, pp. 3799–3802, 2021.

[28] T. Crainic, "Parallel Metaheuristics and Cooperative Search," in *Handbook of Metaheuristics* (M. Gendreau and J.-Y. Potvin, eds.), vol. 272, pp. 419–451, Cham: Springer International Publishing, 2019. Series Title: International Series in Operations Research & Management Science.

[29] G. Parlier, F. Liberatore, M. Demange, and C. a. C. Institute for Systems and Technologies of Information, eds., *ICORES 2019: proceedings of the 8th International Conference on Operations Research and Enterprise Systems: Prague, Czech Republic, February 19-21, 2019*. Setúbal, Portugal: SCITEPRESS - Science and Technology Publications, Lda, 2019. Meeting Name: ICORES.

[30] Y. Kong, "An iterative local search based hybrid algorithm for the service area problem," *Computational Urban Science*, vol. 1, p. 19, Dec. 2021.

[31] D. K. Sharma, R. Pamula, and D. S. Chauhan, "A hybrid evolutionary algorithm based automatic query expansion for enhancing document retrieval system," *Journal of Ambient Intelligence and Humanized Computing*, Feb. 2019.

[32] I. Khennak, "An accelerated PSO for query expansion in web information retrieval: application to medical dataset,"

[33] "Institut national de recherche en informatique et en automatique," *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, vol. 37, pp. 55–57, Dec. 1992.