DATA SCIENCE JOURNAL

Jiang, L et al 2015 The Development of a Combined Search for a Heterogeneous Chemistry Database. *Data Science Journal*, 14: 3, pp.1-9, DOI: http://dx.doi.org/10.5334/dsj-2015-003

PROCEEDINGS PAPER

The Development of a Combined Search for a Heterogeneous Chemistry Database

Lulu Jiang[1,2], Yuehong Zhao[1], Bojun Xu[1] and Hao Wen[1]

[1] State Key Laboratory of Multiphase Complex Systems, Institute of Process Engineering, Chinese Academy of Sciences, Beijing 100190, P R China
yhzhao@ipe.ac.cn

[2] University of the Chinese Academy of Science, Beijing 100049, P R China

A combined search, which joins a slow molecule structure search with a fast compound property search, results in more accurate search results and has been applied in several chemistry databases. However, the problems of search speed differences and combining the two separate search results are two major challenges. In this paper, two kinds of search strategies, synchronous search and asynchronous search, are proposed to solve these problems in the heterogeneous structure database and the property database found in ChemDB, a chemistry database owned by the Institute of Process Engineering, CAS. Their advantages and disadvantages under different conditions are discussed in detail. Furthermore, we applied these two searches to ChemDB and used them to screen for potential molecules that can work as CO_2 absorbents. The results reveal that this combined search discovers reasonable target molecules within an acceptable time frame.

Keywords: Combined search; Synchronous search; Asynchronous search; Chemistry database

1 Introduction

Chemistry databases, one of the primary applications of chemoinformatics (Brown, 2005), are widely used and are becoming a powerful tool (Miller, 2002). Their efficiency and accuracy offer the chemistry researcher a convenient platform with which to acquire target chemistry information. The traditional compound retrieval is a basic simple search (Bunin, Bajorath, Siesel, & Morales, 2006) queried by a single term such as a formula, the numerical limitation of a property, a structural characteristic, and so on. However, these simple searches do not meet users' needs, especially when the database contains millions of records. In many cases, users attempt to query with multiple limits and need to visit several databases at the same time in order to obtain accurate results. This type of retrieval is called an advanced search or a combined search. Because an advanced search obtains a results set by scanning heterogeneous databases, i.e., a structure database and a property database, how to deal with the search speed divergence of different databases is an inevitable problem. Additionally, the efficiency and accuracy of the advanced retrieval is another problem that needs to be considered. Meanwhile, how to combine the separate search result sets is also a challenge.

Currently, the design and development of the advanced search has had some successes. For example, the PubChem Project (http://pubchem.ncbi.nlm.nih.gov/), an open online chemistry database served by the National Center for Biotechnology Information (NCBI), supports an online advanced search that gathers information about molecular structure, properties, chemical elements, and data sources. ChemSpider (http://www.chemspider.com/), another open chemistry database owned by the Royal Society of Chemistry, also supplies a similar advanced search function using a query that combines molecular structure, properties, identifiers, and other filter limitations.

ChemDB (http://www.chemdb.csdb.cn/) is a free online chemical database integrating 24 chemistry databases maintained by 3 institutes of the Chinese Academy of Sciences. It supplies integrated information about a target compound, comprised of its basic properties, thermochemical data, mass spectrum information, and 13 other kinds of data. Currently, its available search methods include a basic search queried by

the compound's name, CAS number, and molecular formula, a molecular structure search, and a compound property search. However, because of the speed with which the number of data records is increasing, the current compound retrieval method cannot satisfy users' requirements. Locating the target compound in the long list of results is extremely time consuming. Therefore, it should be very helpful to develop a new search method to obtain a more exact and explicit results set.

With this aim, the advanced search combining the compound property search and the structure search is a good choice. However, the ChemDB database used for querying includes two heterogeneous databases, the Molecule Structure Database and the Basic Property Database, which each provide an independent search service. Following is a brief introduction of the two databases. The Molecule Structure Database currently contains about 400 thousand records of compound structure information and is constructed on Oracle DBMS with OrChem (Rijinbeek & Steinbeck, 2009) (http://orchem.sourceforge.net/index.html) recording and integrating the molecular structure information primarily in the form of Molefiles (Dalby, Nourse, Hounshell, Guesherst, Grier, Leland, & Laufer, 1992), supporting the structure query based on SMILES input (Weininger & June, 1998). The Basic Property Database currently has collected more than 400 thousand records concerning compound property information and is built on the Microsoft SQL Server. It includes molecular CAS information, molecular weight, formula, etc. However, the problem is that these two heterogeneous databases show a significant difference in their retrieval efficiency. By submitting a certain query for a compound property retrieval, we obtain a result set with 219,172 records in less than 1 second, but it takes more than 7 minutes for a substructure search to attain a satisfactory result set with 255,731 molecular information records. Obviously, the speed gap between the two searches is a challenge when combining these two single search processes. In order to solve this problem, in this paper, we put forward two kinds of schemes, a synchronous search strategy and an asynchronous search strategy.

The rest of the paper is organized as follows. In Section 2, the design and development of the combined search strategies are introduced in detail. The experiment and results are presented in Section 3, and a specific application for the combination search is described in Section 4. Finally, the conclusions are given in Section 5.

2 The Combined Search Design and Development

As introduced above, the combined search needs to achieve a final result set by combining structure search results and property search results. Basically, there are two ways to get this target set. One is to first obtain separated result sets and then find their intersection. The other is to first search one database and utilize this result to scan the other database. However, the significant speed gap between the two databases greatly influences search efficiency and is the most challenging problem in the design. To solve this problem, we offer two possibilities. For the first method, shortening the waiting time brought by the search speed gap can raise search efficiency to a certain degree. For the second method, narrowing the scanning scope can be a useful way to ameliorate the search performance. Correspondingly, two kinds of schemes, synchronous search and asynchronous search, are put forward. The following subsections introduce these searches.

2.1 The synchronous search strategy

The main idea of synchronous search strategy is to obtain results through two steps, first getting separate results sets from the Molecular Structure Database and the Compound Property Database and then finding their intersection. The word "synchronous" means that when a server receives a query message from a browser, it handles the message by visiting the two independent databases simultaneously. However, this simple serial design for the two steps of the synchronous strategy does not perform efficiently because the speed gap between the two heterogeneous databases causes a long wait for the slower search. Thus, for the synchronous search strategy, the key to improving search efficiency is to use parallel operations to shorten the waiting time. On this point, the design for the synchronous search strategy is proposed.

The synchronous search strategy utilizes *queues* to organize the search results from the two individual databases. Meanwhile, the List, a container class provided by language C#, is employed to preserve the combination search results; its element's data type is HashTable.

The synchronous search strategy process is described as follows and in **Figure 1**.

First, users submit a combined query through the browser to the server. The server accepts and processes the query into two SQL queries. They each request only the ID number of the satisfactory compounds. The server utilizes the two SQL queries to visit the corresponding databases simultaneously and obtains the search results, storing them in assigned queues (**Figure 1a**).

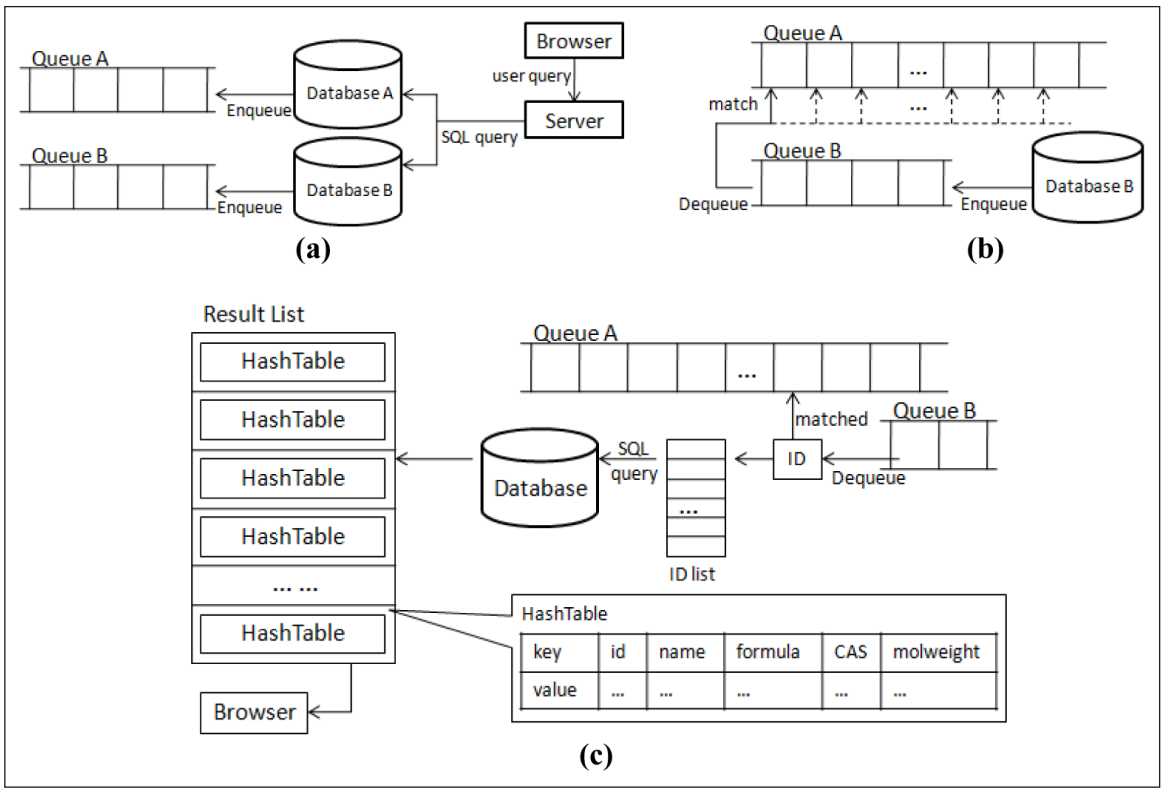


Figure 1: The design of the synchronous search strategy.

During the search process illustrated above, a special thread is assigned to the current combined search as a monitor. Its main duty is to check the status of the two independent searches and pick up the slower one to take to the dequeue operation. As shown in **Figure 1b**, the special thread checks that *queue A* has completed its search task and all satisfactory results have been inserted into *queue A*. At this point, the front element of *queue B* goes to the dequeue operation and compares it with the elements from *queue A*. If the element dequeued from *queue B* exists in *queue A*, its ID will be pushed into the temporary result list (*ID list*) shown in **Figure 1c**; otherwise, it will be thrown out. When the first match operation is completed, the server loops to reexecute the operation described above until the search of Database B is accomplished and meanwhile *queue B* is empty. Finally, the process utilizes the *ID List* to revisit the databases to obtain the complete information for the queried compounds and preserve it in the *Result List* in **Figure 1c**.

In a synchronous search process, the database is visited three times: The two single searches plus a third visit to obtain the complete information of the elements in the *ID List*. Only when the combined search results set is empty does a synchronous search visit a database twice. As an obvious advantage, the relatively fixed frequency of database visits guarantees the efficiency of the combined search to a certain degree. However, its weakness cannot be ignored. In a synchronous search process, both of the individual searches need to scan the whole database no matter how many results the user requests. Additionally, the intersection process of a synchronous search is a time consuming operation compared with an asynchronous search. The specific discussion of this is described in Section 3.

2.2 Asynchronous search strategy

As opposed to the synchronous search strategy, the asynchronous search strategy is concerned with the search sequence. It visits one database first and then utilizes the results as the search limit with which to scan the other database. Taking full advantage of the current results to do the next search significantly narrows the scope of the scan. Meanwhile, as the structure search is a time consuming process, the combined search efficiency will be ameliorated if the waiting time for the structure results can be utilized. Because of this, the structure search is chosen to be operated first in the design. Once the structure search gets a specific quantity of results, the property search will be activated and the structure results will then be used to complete the combined search. The word “asynchronous” here indicates that the visits to the two separated databases are sequenced rather than parallel.

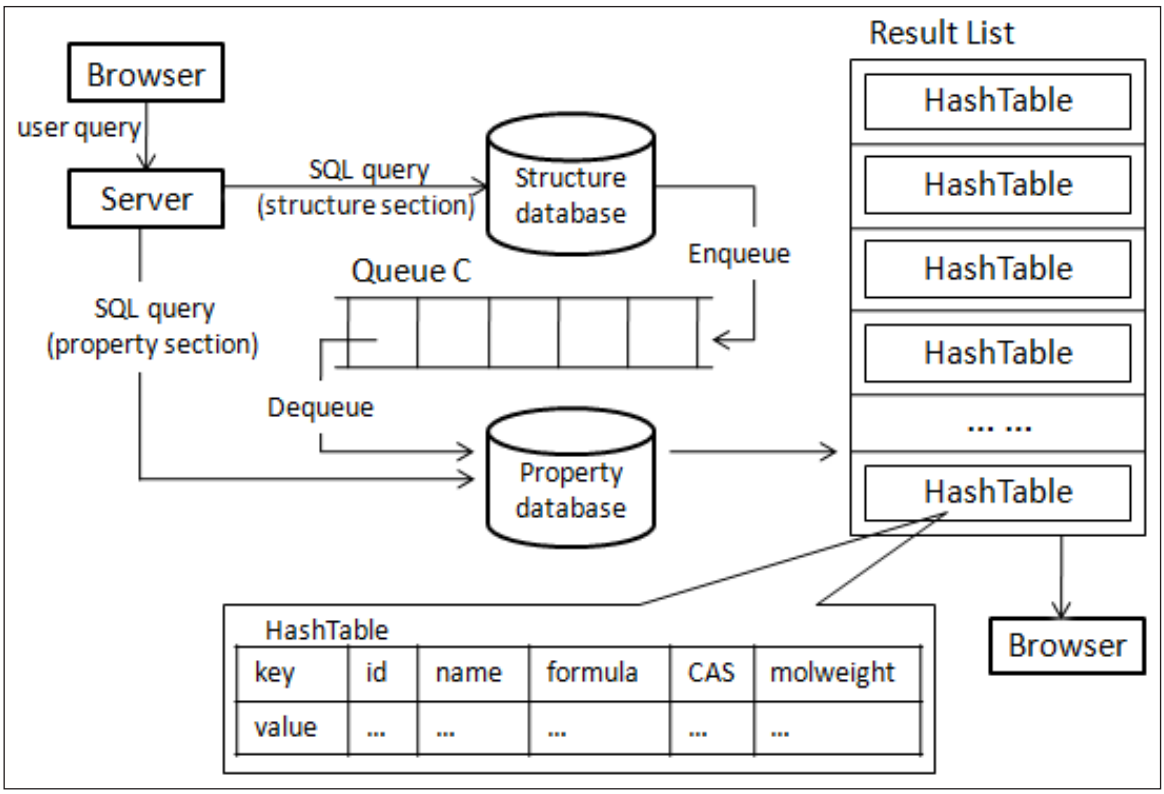


Figure 2: The design of the asynchronous search strategy.

In an asynchronous search process, a *queue* is used to preserve the molecular structure search results. A List, the same as in the synchronous strategy, is used to organize the final combined search results set; its element is HashTable.

The design of the asynchronous search strategy is shown in **Figure 2**.

First, the server accepts the query from the user, extracting the structure search section and then putting it into a piece of SQL string. Then the SQL string is used to scan the Molecular Structure Database for the asked for results. The results, which only contain the ID information of the compounds, are maintained in *queue C* as illustrated in **Figure 2**.

Then the front element of the obtained *queue C* is dequeued and combined with the property limitation submitted from the browser to be spelled out in a new SQL query. Next, the server connects to the Compound Basic Property Database with the specified SQL query to get the asked for result and adds it to the *Result List* noted in **Figure 2**. Once the operation is completed, the server continues to get the new front element of *queue C* and connects to the database again. The operation is repeated as above until no elements are left in *queue C*. However, this is a very time-consuming action if one operation cycle only handles a single element because it needs to connect to the database too many times. Therefore, it is essential to convert the single element work into batches. In this paper, we set the once dequeue element number to be 2000, which reduces connection times and eventually significantly improves the efficient of the combination search as discussed below.

Compared with the synchronous search strategy, the connection time of an asynchronous search is not fixed. Here we set the result numbers to be obtained from the single structure search to be *N*, the number of dequeued elements in queue C to be σ, and the number of times to connect to the databases to be *n*. From these we get the formula: $n = \left\lfloor \frac{N}{\sigma} \right\rfloor + \left\lceil \frac{N \% \sigma}{\sigma} \right\rceil$. According to this formula, the larger we set σ, the smaller the *n*. However, it is impossible to set σ to an infinite size. Therefore, we set parameter σ to 2000 in consideration of the system's capability. In different situations, *n* should be set to different values. This service will only visit the database once when $n = 1$; if $N \leq \sigma$, then *n* will equal 2; but if $N > \sigma$, then *n* cannot be specifically defined. In most cases, however, $N > \sigma$ and $n > 3$, therefore compared with the synchronous search schema, the frequency of asynchronous search visits to the databases is much greater. On the other hand, an asynchronous search scans the whole database only once, a smaller number than that in a synchronous search operation.

3 Experiments and Results

In order to compare the efficiency of synchronous searches with asynchronous searches under different conditions, we designed some experiments, as below. Also, we present a discussion of key factors affecting search efficiency.

3.1 Experiment design and environment

To cover all retrieval circumstances, we selected a test sample containing 45 queries to check, 4 of which are listed in **Table 1**. Each query consists of a structure limit section and a property limit section. Based on this, the experiment was designed to use each query to get 3 temporal data records. The amount of time taken to obtain results sets with 20, 50, and 100 records was recorded for further analysis. The query design is in accordance with the following principles:

First, control the scale of the structure search results. The experiment was designed to employ different substructure queries to cover a variety of structure search results. Twenty-five types of substructure were chosen while the scale of the search results varied from 250 thousand to 20 records.

Second, control the scale of the property search results. A single property search of ChemDB supplied 14 kinds of limits to choose. The design assembled some of the limits and picked up 15 property query records as a test sample. The number of search results ranged from 219,172 to 1,522 records.

Third, the experiments should balance the results scale of the structure search with that of the property search. The 45 test records contained 3 types of queries, a big structure set with a small property set, a small structure set with a big property set, and structure and property sets of equal size; this includes all possible kinds of query circumstances.

Last, the experiment sample should include several controlled test sets in order to explore the factors influencing search efficiency. Because the similar structure search and substructure search differ in their target sets and search measures, it is meaningless to compare their search efficiency. Therefore, our experiment only records the results of the substructure search for analysis.

The whole experiment is based on the environment as follows. The Molecular Structure Database is constructed on the Oracle 11g R2(x64) Edition, and the Basic Property Database is built on the Microsoft SQL Server 2005 Edition. Both DBMSes referred to in the experiment are constructed on a computer with Intel(R) Xeon(R) CPU 5140 and based on the Microsoft Windows Server 2003 R2(x64) operating system. Also, the server computer provides service with the Inter(R) Core(TM) I5–3470 CPU, and the system operation edition is Windows 7 Service Pack 1(x86).

3.2 Results and discussion

The experiment tested 45 combined queries using two combined search strategies and obtained the test results shown in **Figure 3**. In conclusion, the asynchronous search performed better than the synchronous search in many cases.

Moreover, we recorded the number of single search results and plotted them with the time-cost data in **Figure 4**. The time data shown in **Figure 4** records the time cost of a query asking for 100 result records.

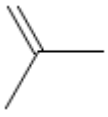
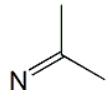
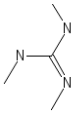
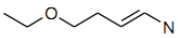
NO.	Structure Limit	Search Type	Property Limit	Result Number
1		substructure	Molar Mass [0,200] Formal Charge [0,0] Solubility(logS) S:mol/L [-1,1]	100/50/20
2		substructure	Molar Mass [0,200] Formal Charge [0,0]	100/50/20
3		substructure	Molar Mass [300,500] Formal Charge [0,0] Number of Hydrogen Bond Ligand [1,20]	100/50/20
4		substructure	Molar Mass [100,300] Formal Charge [0,0] Solubility(logS) S:mol/L [-1,1]	100/50/20

Table 1: The partial combination queries for the comparison experiment.

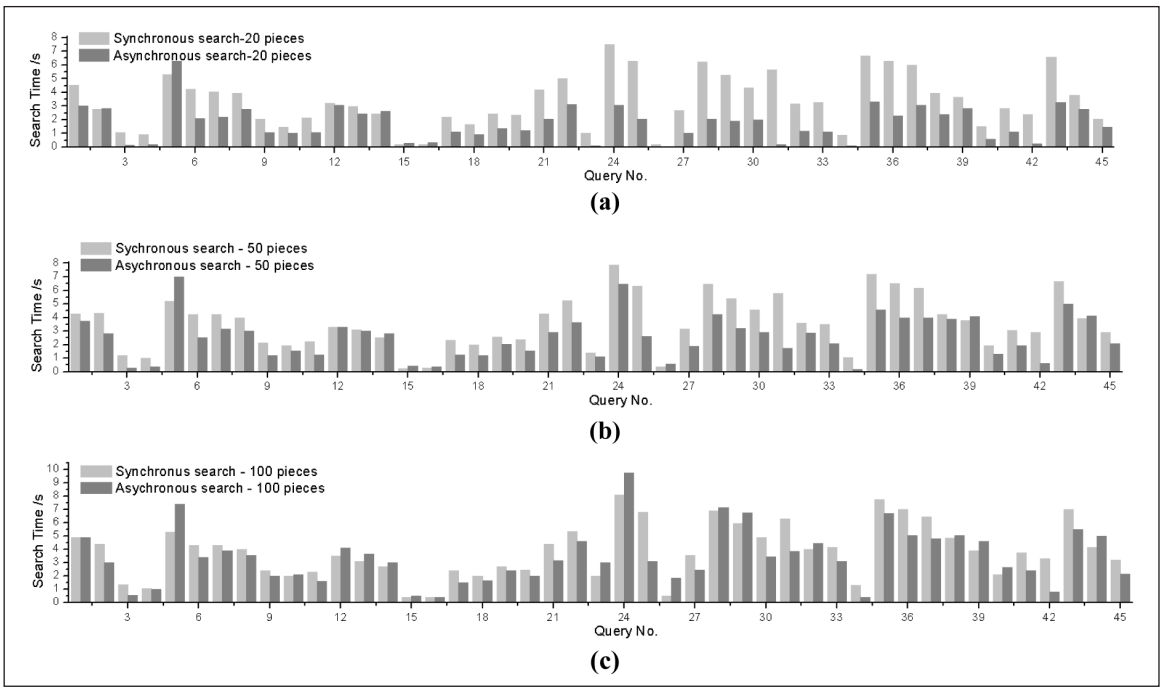


Figure 3: Test statistics of the queries. **(a), (b), (c)** show the performance when the search returns 20, 50, 100 result records.

According to an analysis of the results, we can conclude that the following factors influenced search efficiency.

First, the scale of structure search results, that is, the time-cost of the structure search has a great effect on the time-cost of the combined search process for both synchronous and asynchronous searches. According to the results displayed in **Figure 4**, from query No.1 to No. 4, the number of structure search results decreased greatly and the combined search efficiency increased correspondingly. Compared with the structure search, the variety of property search results quantities has little influence on the combined search efficiency. The main reason for this is that the structure search itself is a time consuming operation.

Second, limiting the number of return results greatly influenced synchronous search efficiency. In contrast, it basically has no influence on the synchronous search process. According to the data shown in **Figure 3**, it is obvious that increasing the returning records limit makes the time-cost of the synchronous search vary greatly but does not affect the asynchronous search. The time-cost is determined by the search strategy itself. As discussed in Section 2, the frequency of database connections in an asynchronous search process is not defined. When the number of returned records increases, there is a great probability that connection frequency also increases. However, in a synchronous search process, the connection frequency is basically fixed (discussed in Section 2.1), which means that it performs steadily even though the limit on the amount of results changes. In other words, the synchronous search will be superior when the query needs to obtain all satisfactory search results. The experiments have also corroborated this point.

Furthermore, the efficiency of the intersection operation is another factor affecting search performance. According to the introduction of Section 2, under complete results search conditions, the time complexity of a synchronous search is *O(MN)*, where M stands for the number of property search results and N refers to the number of structure search results. As for an asynchronous search, it only consumes *O(N)* time complexity to complete a search process. In this regard, the asynchronous search performs better.

In summary, the scale of the structure results greatly influences both synchronous and asynchronous searches while the efficiency of the asynchronous search is influenced by the number of requested results.

4 Application

As mentioned in the introduction, a combined search is widely used in chemistry database systems. To verify the accuracy of these search results, our team utilized the combined search introduced above to generate an explicit candidate set for the further molecular design of CO_2 absorbents. The combined search greatly narrowed the research range and helped us focus efficiently on the target set while the generated candidate molecule lay the foundation for the following screening work.

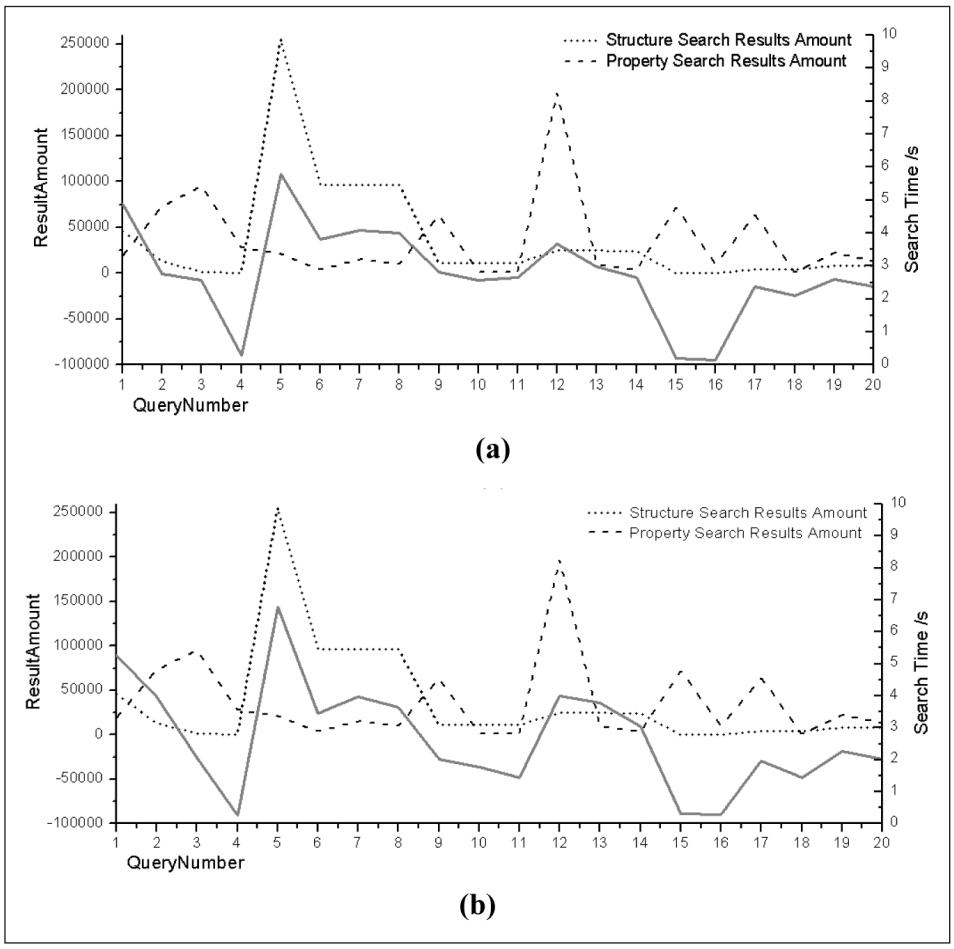


Figure 4: The statistics of the combined search for 100 results records. (a), (b) respectively depict the performance of the synchronous search and the asynchronous search.

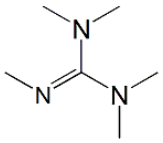
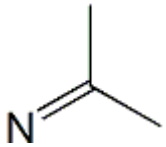
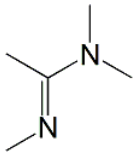
Property limitation	Query structure	Result amount
		30
Molar mass(g/mol) [0,150] Solubility(logS) S:mol/L [0,10]		15
		58

Table 2: The partial queries for design of the CO_2 absorbents.

In our verification, we analyzed specific design requirements, proposing a corresponding query to limit the results set. The experiment took the active groups as query substructures and physicochemical characteristics as constraint conditions (Wang, 2014). Specifically, the work chose the nitrogenous groups as query substructures and limited the compound solubility to greater than 1 mol/L and the molar mass to less than 150g/mol. Some of combined queries are listed in **Table 2**. The experiment obtained a total of

498 compounds with all the queries. After a preliminary judgment using our knowledge of chemistry, the generated candidate molecules were roughly certified to be correct.. Finally our team picked 10 molecules for further design and experimentation.

5 Conclusion

In this paper, two schemes for combined searches were proposed. According to the statistics obtained from the experiment, we explored factors influencing search efficiency and discussed their influences in different situations. First, the structure results set is a primary factor affecting the time-cost of both search strategies. Next, limiting the number of results affects an asynchronous search but has no effect on a synchronous search. This is the result of the frequency of connections to the database. Moreover, the asynchronous search has a smaller time frame in which it obtains the final results sets. Therefore, we conclude that the asynchronous search is much more appropriate for ChemDB. However, in comparison with a synchronous search, an asynchronous search lacks stability, as shown by the experimental data. Thus balancing time-cost with stability needs further research in the future. Finally, we applied the combination search to discover the molecular design of CO_2 absorbents. By analysis of the search results, we demonstrated the correctness of the combined results set.

6 Acknowlegements

The work was financially supported by the Scientific Data Resource Integration and Sharing Project, Chinese Academy of Science, XXH12504–1–07.

7 References

Brown, F. (2005) Editorial Opinion: Chemoinformatics – a ten year update. *Current Opinion in Drug Discovery & Development 8*(3), pp 296–302.

Bunin, B. A., Bajorath, J., Siesel, B., & Morales, G. (2006) *Chemoinformatics: Theory, Practice, & Products,* Springer.

Dalby, A., Nourse, J. G., Hounshell, W. D., Guesherst, A. K., Grier, D. L., Leland, B. A., & Laufer, J. (1992) Description of several chemical structure file formats used by a computer program developed at Molecule Design Limited. *Journal of Chemical Information and Computer Sciences 32*, pp 224–225.

Miller, M. A. (2002) Chemical database techniques in drug discovery. *Nature Reviews Drug Discovery 1*, pp 220–227.

Rijinbeek, M. & Steinbeck, C. (2009) OrChem-an open source chemistry search engine for Oracle. *Journal of Chemical Information and Computer Sciences 1*(17), p 2.

Wang, Y. Q. (2014) *Candidate Generation and Screening for Guanidine and Amidine CO2 Absorbents*, PhD thesis, University of the Chinese Academy of Science, Beijing, P R China.

Weininger, D. & June, R. (1998) SMILES: A Chemical Language and Information System. *Journal of Chemical Information and Computer Sciences 28*(1), pp 31–36.

How to cite this article: Jiang, L, Zhao, Y, Xu, B and Wen, H 2015 The Development of a Combined Search for a Heterogeneous Chemistry Database. *Data Science Journal,* 14: 3, pp.1-9, DOI: http://dx.doi.org/10.5334/dsj-2015-003

Published: 22 May 2015

]u[*Data Science Journal* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS