On Large-Scale Qualitative Spatio-Temporal Constraint Redundancy Removal

1st Qiyuan Hu
School of Computing and Artificial Intelligence
Southwest Jiaotong University
Chengdu, China
huqiyuan@my.swjtu.edu.cn

2nd Yang Gao
School of Computing and Artificial Intelligence
Southwest Jiaotong University
Chengdu, China
2415383726@my.swjtu.edu.cn

3rd Zhiguo Long*
School of Computing and Artificial Intelligence
Southwest Jiaotong University
Chengdu, China
zhiguolong@swjtu.edu.cn

4th Hongjun Wang
School of Computing and Artificial Intelligence
Southwest Jiaotong University
Chengdu, China
wanghongjun@swjtu.edu.cn

5th Tianrui Li
School of Computing and Artificial Intelligence
Southwest Jiaotong University
Chengdu, China
trli@swjtu.edu.cn

6th Michael Sioutis
Faculty of Information Systems and Applied Computer Sciences
University of Bamberg
Bamberg, Germany
michail.sioutis@uni-bamberg.de

***Abstract*—Qualitative spatio-temporal representation and reasoning is an important research direction in the field of artificial intelligence, and qualitative constraint networks are a key tool for representing and reasoning with spatio-temporal knowledge. Previous studies have pointed out that qualitative constraint networks often contain redundant constraints. This not only results in a waste of storage and transmission resources, but can also be a bottleneck in various algorithms and applications that are based on qualitative constraint networks. Existing research proposed several algorithms to deal with redundant constraints, but their time complexities are relatively high, which is very unfavorable when dealing with large-scale data that are common nowadays. This paper proposes a new model based on distributed computation to deal with redundant constraints efficiently. Specifically, this paper theoretically proves the feasibility of the distributed model for this problem, proposes different optimization algorithms for distributed task allocation, and evaluates the performance of the distributed model through a variety of experiments on real-world data sets. The results show that the distributed model can significantly improve the processing efficiency of simplifying representations of qualitative spatio-temporal information on large-scale data sets, and that the distributed solution to the problem of simplifying representations can also improve data privacy, which can contribute to big data processing and data security protection in qualitative spatio-temporal reasoning research.**

***Index Terms*—qualitative spatio-temporal representation and reasoning, qualitative constraint network, representation simplification, redundancy, distributed algorithm**

I. Introduction

Spatio-temporal information is very important in both daily life and scientific research. For example, people often need to schedule different tasks (such as appointments), and it is necessary to model the spatial relations between surrounding objects in robot control and navigation [1], like “the cup is above the table” and “the hand of the robot is next to the cup”. This type of information can be processed by qualitative spatio-temporal representation and reasoning (QSTR). QSTR is an important sub-field of artificial intelligence, and plays an invaluable role in related fields, such as image processing [2], computer vision understanding [3], and deep learning [4]. In QSTR, qualitative spatio-temporal information is usually represented by a graph structure, called *qualitative constraint network* (QCN), in which the vertices of the graph correspond to spatial or temporal objects, the edges to the connections between these objects, and the label of an edge to a qualitative relation between the corresponding two objects. As shown in Fig. 1, the QCN on the right represents the relation (constraint) between the spatial objects on the left. For example, the relation between 1 and 2 is **PO** (partially overlap).

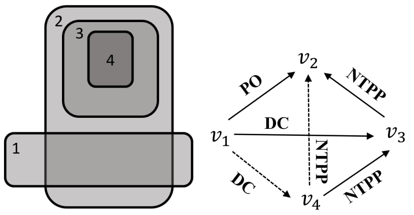

Fig. 1. Illustration of QCN and redundant relations.

Note that a QCN for n objects needs $\mathcal{O}(n^2)$ edges to represent the relations. For large n, this representation will not be favorable to storage and transmission, as well as to algorithms that use QCNs for qualitative information processing. For example, Wallgrün [5] proposes to adjust spatial regions to meet some given topological constraints. The efficiency of the adjustment algorithm is in negative proportion to the number of topological constraints, which means that fewer constraints will lead to higher efficiency of the algorithm. The number

This work was supported by the National Natural Science Foundation of China (61806170), the Humanities and Social Sciences Fund of Ministry of Education (18XJC72040001), and the National Key Research and Development Program of China (2019YFB1706104). *Corresponding author: Zhiguo Long.

of constraints will also affect reasoning efficiency. When checking if some spatial objects satisfy certain qualitative constraints, fewer constraints means fewer checks in general. Smaller number of constraints is also helpful in comparing two QCNs. For example, in the merging problem of QCNs [6], it is often necessary to calculate the distance between two QCNs, fewer constraints means faster calculation of the distance. From the above, it can be seen that simplifying the representation of a QCN has important practical advantages, and identifying and removing *redundant constraints* is important. Redundant constraints refer to constraints in a QCN whose removal does not change the represented information. Such constraints are very common in QCNs. For example, in Fig. 1, the constraint between v_4 and v_2 can be entailed by the constraints between v_4 and v_3, and between v_3 and v_2 (4 is strictly contained in 3, 3 is strictly contained in 2, and thus 4 is strictly contained in 2), i.e., this constraint is redundant. Similarly, the constraint between 1 and 4 is also redundant.

In this paper, we propose to combine distributed algorithms to solve the redundancy problem, which can also ensure efficiency and privacy. The main contributions of this paper are as follows:

- we theoretically prove the feasibility of establishing a distributed model for removing redundant constraints of large-scale QCNs, by showing that the model can correctly identify redundant constraints, and can remove these redundant constraints in a distributed manner;
- we construct a distributed model for removing redundant constraints of large-scale QCNs, devise the corresponding distributed algorithm, propose several task allocation optimization algorithms, and theoretically demonstrate their effectiveness;
- we use three baseline algorithms to conduct comparative experiments on 17 real-world data sets, where the results show the superiority of the proposed distributed model and optimization algorithms, and the significance of performance improvement.

The remainder of this paper is structured as follows: Section II discusses related work, Section III introduces preliminary knowledge and notions, Section IV proposes and analyses the distributed model, algorithm, and optimizations, Section V performs experimental evaluations, and Section VI concludes the paper.

II. Related Work

QSTR has been extensively studied in the past decades. Researchers considered models and representations of qualitative spatio-temporal information [7]–[10], efficient reasoning with such information [11]–[14], and its applications in different areas [1], [2], [15], [16]. The redundancy of knowledge has already attracted the attention of researchers in various fields [17]–[20]. In QSTR, Egenhofer and Sharma were the first to consider the redundancy problem [21], by observing that some constraints in a given QCN were not necessary for answering queries. This problem was again mentioned by Rodríguez et al. [22]. Wallgrün [5] found that redundant constraints have a great impact on the performance of topological adjustment algorithms, and for the first time, gave two algorithms for removing redundancy in QCNs, which however have no guarantee to remove all redundant constraints. Recently, Li et al. [23] comprehensively studied the redundancy problem of QCNs from the perspective of relation algebras, and based on special relational subclasses (i.e. distributive subalgebras [24]) proposed a complete algorithm that can remove all redundant constraints. Following that work, Sioutis et al. [25] proposed a redundancy removal algorithm for sparse QCNs. However, the worst time complexity of these two algorithms is $\mathcal{O}(n^3)$ (n is the number of variables in the constraint network). For the case when there are a large number of variables and constraints, they will naturally be inefficient. Moreover, these two algorithms are based on a centralized processing model, resulting in a waste of computation power and a loss of data privacy. On the other hand, the distributed model proposed in this paper can make use of current popular distributed computation frameworks to perform efficient calculations, so that the idea of removing redundancy can be applied efficiently to large-scale data, and to enhance the ease of use of these data and also protect data privacy and security.

There are some precedents for distributed algorithms in the field of qualitative spatio-temporal representation and reasoning [26]–[29]. These distributed algorithms are only for consistency checking problems. For the redundancy problem, although there is various research, distributed related algorithms are rarely studied. Therefore, this paper also plays an important role in promoting the application of distributed algorithms to the redundancy problem.

III. Preliminaries

Qualitative relations are the most fundamental concept in QSTR. Given a domain D, a (binary) relation R on D is a set of pairs $(a, b) \in D \times D$, i.e., $R \subseteq D \times D$. In particular, some set of relations can form a partition B of $D \times D$, where for any pair $(a, b) \in D \times D$, there is a unique relation $r \in B$ such that $(a, b) \in r$. This partition B, together with set intersection, union, and complement, gives a particular Boolean algebra, which is known as a *qualitative calculus*. Different partitions B give different qualitative calculi, and a relation r in B is called a *basic (qualitative) relation*. The Region Connection Calculus (RCC) [30] is a classic model that characterizes topological relations between spatial regions and has applications in various fields [4], [31]. As shown in Fig. 2, RCC has a specific model called RCC8, which contains eight basic relations.

A relation obtained by the intersection, union, and complement of the basic relations is still a qualitative relation in the qualitative calculus, such as $\mathbf{DC} \cup \mathbf{PO}$. For convenience, we usually use a set $\{r_1, \ldots, r_k\}$ to represent a qualitative relation $r_1 \cup \cdots \cup r_k$, such as $\{\mathbf{DC}, \mathbf{PO}\}$. When $r_1 \cup \cdots \cup r_k = D \times D$, this qualitative relation is called the *universe relation*, denoted as $\mathbf{U}$; when $k = 1$, for representation consistency, we sometimes also use $\{r_1\}$ to represent the basic relation r_1.

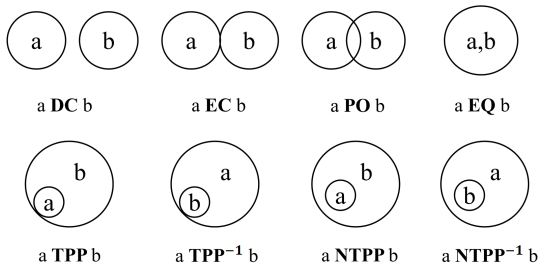

Fig. 2. Illustration of RCC8 relations.

The inverse relation of a relation R is written as R^{-1}, such as the inverse relation of $\mathbf{TPP}$ is $\mathbf{TPP}^{-1}$. The identity relation is denoted by $\mathbf{EQ}$.

Qualitative relations in a qualitative calculus can be composed in a usual way as general relations: $R \circ S = \{(x, y) | \exists z \in D, (x, z) \in R \wedge (z, y) \in S\}$, However, the result of the composition is not necessarily a qualitative relation in the qualitative calculus. Therefore, researchers proposed the concept of *weak composition*, which is defined as $r_1 \diamond r_2 = \{r \in B \mid r \cap (r_1 \circ r_2) \neq \emptyset\}$, i.e., the smallest qualitative relation that contains the usual composition. The weak composition between two general qualitative relations is defined as $\bigcup \{r_1 \diamond r_2 \mid r_1 \in R_1, r_2 \in R_2\}$.

Definition 1. Given a qualitative calculus $\mathcal{M}$, if a set of qualitative relations $\mathcal{S} \subseteq \mathcal{M}$ satisfies the following simultaneously:

- $B \subseteq \mathcal{S}$;
- $\mathcal{S}$ is closed under the intersection of relations, weak composition, and inverse operations;
- For any three relations $R, S, T \in \mathcal{S}$, if $S \cap T \neq \emptyset$, then $R \diamond (S \cap T) = (R \diamond S) \cap (R \diamond T)$ and $(S \cap T) \diamond R = (S \diamond R) \cap (T \diamond R)$;

then $\mathcal{S}$ is a *distributive subalgebra* of $\mathcal{M}$.

Qualitative spatio-temporal information is usually represented by qualitative constraint networks.

Definition 2. A *qualitative constraint network* (QCN) N is an pair (V, C), where V is a set of variables, C is a set of constraints of the form $(v_i R_{ij} v_j)$, R_{ij} is a qualitative relation in the qualitative calculus model $\mathcal{M}$, and for all $v_i, v_j \in V$ we have $R_{ij} = R_{ji}^{-1}$, and $R_{ii} = \mathbf{EQ}$.

For convenience, we sometimes use $C(v_i, v_j)$ or $N[v_i, v_j]$ to represent the relation R_{ij} between v_i and v_j. If there is an explicit constraint between each pair of variables in a QCN, then we call it a *complete network*, otherwise it is called an *incomplete network*. If the relations in a complete network are all basic relations, then it is called a *basic network*. As shown in Fig. 1, the relations in this QCN are all basic relations (some inverse relations are omitted in the figure), and it is a basic network; but if the relations corresponding to the dashed edges were removed, the resulting network would be an incomplete network.

Definition 3. A QCN $N = (V, C)$ naturally induces an undirected graph $G(N) = (V, E_C)$, that is, the vertex set of $G(N)$ is V, and an edge $(v_i, v_j) \in E_C$ if $R_{ij} \neq U$ and $v_i \neq v_j$. This undirected graph is called the *constraint graph* of N.

Given a QCN N, $N_{[v_i,v_j]/R}$ means replacing the relation between v_i and v_j by R, i.e., $N_{[v_i,v_j]/R} = (V, C')$, where $C'(v, v') = R, C'(v', v) = R^{-1}$, and for any $(x, y) \in V \times V - \{(v, v'), (v', v)\}$, $C'(x, y) = C(x, y)$.

Definition 4. A *solution* a of QCN $N = (V, C)$ is an assignment that maps all the variables in V into the domain D, such that $\forall v_i, v_j \in V$, $(a(v_i), a(v_j)) \in N[v_i, v_j]$. If N has such a solution, then we say N is *consistent* or *satisfiable*. If two QCNs N_1 and N_2 have the same solution set, then these two QCN are *equivalent*, denoted by $N_1 \equiv N_2$. If there are no two different variables v_i and v_j of N such that for any solution of N, $(a(v_i), a(v_j)) \in \mathbf{EQ}$, then N is said to satisfy the *uniqueness property* (because every variable corresponds to a unique entity); clearly, every QCN can be made to satisfy this property by collapsing identical entities together.

To determine the consistency of a QCN, researchers have proposed many algorithms, among which partial path consistency is a commonly used criterion.

Definition 5. Given a QCN $N = (V, C)$, suppose $G(N) = (V, E_C)$ is its constraint graph, $G = (V, E)$ is another graph satisfying $E_C \subseteq E$. If $\forall (v_i, v_j)$, (v_i, v_k), $(v_k, v_j) \in E$, $N[v_i, v_j] \subseteq N[v_i, v_k] \diamond N[v_k, v_j]$ and $N[v_i, v_j] \neq \emptyset$, then N is *partial path consistent* (PPC) w.r.t. G.

A consistent QCN N can always be converted into an equivalent partial path consistent QCN $\diamond_G(N)$ by using the partial path consistency algorithm (PPC) [32], known as the *partial path consistency closure* of N. Researchers found that when a QCN has a special graph structure, reasoning on it can be more efficient. One of the important graph structures is the triangulation graph or the chordal graph.

Definition 6. An undirected simple graph is called a *triangulation graph* or *chordal graph*, if for any of its cycles containing 4 or more edges (i.e., loops without repeated vertices), there is an edge connecting two non-adjacent vertices on this cycle.

If G is a chordal graph, for QCNs over a distributive subalgebra of RCC8, the consistency of N can be determined by PPC [33].

IV. Redundancy Removal Model and Algorithms

A. The Redundancy Problem

As mentioned earlier, there are often redundant constraints in QCNs, and the goal of this paper is to propose an efficient distributed algorithm to identity and remove such constraints.

Definition 7. Given a QCN $N = (V, C)$, a constraint $(v_i R_{ij} v_j)$ is *redundant* iff $N_{[v_i,v_j]/B} \equiv N$.

In other words, a constraint $(v_i R_{ij} v_j)$ is redundant, if its removal (together with its inverse) does not change the information expressed by the resulting QCN. A natural question is how to check if a constraint $(v_i R_{ij} v_j)$ is redundant. According to the definition of redundancy, this is equivalent to checking that $N_{[v_i,v_j]/B}$ and N are equivalent, that is, for any basic relation r in $B - R_{ij}$, verifying that $N_{[v_i,v_j]/r}$ is unsatisfiable. This leads to the most basic algorithm with time complexity $\mathcal{O}(n^5)$, which is too expensive.

Li et al. [23] proposed a more efficient algorithm, with time complexity $\mathcal{O}(n^3)$, for identifying and removing redundant constraints of a complete network over a distribution subalgebra. However, the requirement of complete network usually cannot be satisfied in practical applications, and thus Sioutis et al [25] later gave a more efficient algorithm for incomplete networks over some distributive subalgebra. Theorem 1 guarantees the correctness of their algorithm.

Theorem 1. *([25]) Let $\mathcal{S}$ be some distributive subalgebra of RCC8, $N = (V, C)$ a QCN on $\mathcal{S}$ satisfying the uniqueness property, $G = (V, E)$ a chordal graph satisfying $E(G(N)) \subseteq E$, and N^* the partial path consistency closure of N w.r.t. G. Then $N[v_i, v_j]$ is redundant in N iff $N^*[v_i, v_j] = \bigcap\{N^*[v_i, v_k] \diamond N^*[v_k, v_j] \mid (v_i, v_k), (v_k, v_j) \in E\}$ or $N[v_i, v_j] = \mathbf{U}$.*

The corresponding algorithm is shown in Algorithm 1 (Prime), in which the redundancy checking sub-procedure is given in Algorithm 2 (CheckRedun). When G is a complete graph, the algorithm degenerates to the algorithm on a complete network.

Algorithm 1: Prime

```
Input: A QCN $N = (V, C)$ over a distributive
       subalgebra of RCC8; a chordal graph
       $G = (V, E)$ satisfying $E(G(N)) \subseteq E$.
Output: $N_c$ without redundant constraints.
1 $N_c \leftarrow N$;
2 $N \leftarrow \diamond_G(N)$;
3 for $(v_i, v_j) \in E$ do
4     $N_c \leftarrow$ CheckRedun($N, N_c, (v_i, v_j)$);
5 end
6 return $N_c$;
```

B. Distributed Redundancy Removal Model and Feasibility Analysis

In order to improve redundancy identification and removal in large-scale data, this paper proposes a distributed model, which is illustrated in Fig. 3.

First, an input QCN will be preprocessed by Preprocess(), including the calculation of its partial path consistency closure; then the redundancy checking of a constraint $N[v_i, v_j]$ in the QCN is made as a single distributed task, and $N[v_i, v_j]$ is passed to different computation nodes by a certain allocation algorithm AllocateTask(); CheckRedun() (Algorithm 2) is then used for redundancy checking and removal on each node; finally, the CollectResult() algorithm is used to collect the

Algorithm 2: CheckRedun

```
Input: QCNs $N$ and $N_c$, and an edge $(v_i, v_j) \in E$ to
       check consistency.
Output: The updated $N_c$.
1 if $N_c[v_i, v_j] = \mathbf{U}$ then
2     $N_c \leftarrow N_c - N[v_i, v_j]$;
3     return $N_c$;
4 end
5 $Q_{ij} \leftarrow \mathbf{U}$;
6 for $v_k$ s.t. $(v_i, v_k), (v_k, v_j) \in E$ do
7     $Q_{ij} \leftarrow Q_{ij} \cap N[v_i, v_k] \diamond N[v_k, v_j]$;
8     if $Q_{ij} = N[v_i, v_j]$ then
9         $N_c \leftarrow N_c - N[v_i, v_j]$;
10        return $N_c$;
11    end
12 end
13 return $N_c$;
```

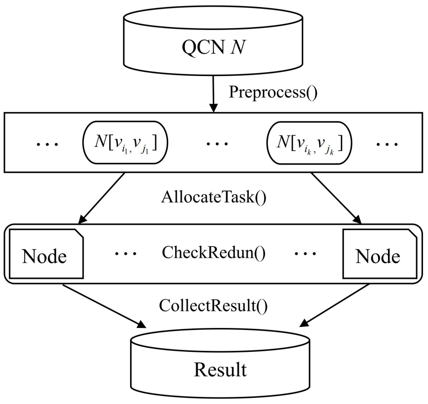
QCN N
Preprocess()
$\cdots$ $N[v_{i_1}, v_{j_1}]$ $\cdots$ $N[v_{i_k}, v_{j_k}]$ $\cdots$
AllocateTask()
Node $\cdots$ CheckRedun() $\cdots$ Node
CollectResult()
Result

Fig. 3. Illustration of the distributed model.

results on all computation nodes. The general flow of the model is shown in Algorithm 3.

The theorem below shows that for a QCN N satisfying the uniqueness condition over a distributive subalgebra of RCC8, N_c returned by Algorithm 3 has no redundant constraint, and is equivalent to N, that is, the large-scale distributed computation model proposed in the paper is sound and complete.

Theorem 2. *Let $\mathcal{S}$ be a distributive subalgebra of RCC8, N a QCN over $\mathcal{S}$ satisfying the uniqueness property, $G = (V, E)$ a chordal graph satisfying $E(G(N)) \subseteq E$, and N^* the partial path consistency closure of N w.r.t. G. If N_c is the returned result of Algorithm 3 with N and G as input, then N_c has no redundant constraint and $N_c \equiv N$.*

Proof. According to the second line of Algorithm 3, the QCN used to determine the redundant constraint in line 7 is the partial path consistency closure of N w.r.t. G, and by Theorem 1, the redundancy of any constraint $N[v_i, v_j]$ allocated to a certain computation node can be determined by Algorithm 2. Therefore, all the redundant constraints in N

Algorithm 3: Distributed Prime

```
Input: A QCN $N = (V, C)$ over a distributive subalgebra of RCC8; a chordal graph $G = (V, E)$ satisfying $E(G(N)) \subseteq E$; the number of computation cores $K$.
Output: $N_c$ without redundant constraints.
1  $N_c \leftarrow N$;
2  $N \leftarrow \diamond_G(N)$;
3  $\{C_1, \cdots, C_K\} \leftarrow$ AllocateTask$(N_c, G, K)$;
4  foreach node r parallelly do
5      $N_c^{(r)} \leftarrow C_r$;
6      for each $(v_i, v_j) \in N_c^{(r)}$ do
7          $N_c^{(r)} \leftarrow$ CheckRedun$(N, N_c^{(r)}, (v_i, v_j))$;
8      end
9  end
10 $N_c \leftarrow$ CollectResult$(N_c^{(1)}, \cdots, N_c^{(K)})$;
11 return $N_c$;
```

can be identified and removed, that is, N_c has no redundant constraints.

Secondly, the order to remove redundant constraints does not affect the redundancy of the constraints, that is, the operations to identify and remove redundant constraints can be parallelized. Suppose $N[x_1, y_1]$ and $N[x_2, y_2]$ are redundant constraints in N. Let $N^{(1)} = N_{[x_1,y_1]/B}$ and $N^{(2)} = N_{[x_2,y_2]/B}$, i.e., $N^{(1)}$ is the resulting QCN when removing $N[x_1, y_1]$ from N and $N^{(2)}$ is the resulting QCN when removing $N[x_2, y_2]$ from N. By the redundancy of $N[x_1, y_1]$ and $N[x_2, y_2]$, the partial path consistency closures of $N^{(1)}$ and $N^{(2)}$ w.r.t. G are both N^*. As the redundancy checking only uses N^* according to Algorithm 2, $N[x_1, y_1]$ is also redundant in $N^{(2)}$ (similar for $N[x_2, y_2]$). Therefore, $N^{(2)}_{[x_1,y_1]/B} \equiv N \equiv N^{(1)}_{[x_2,y_2]/B}$. Then N_c obtained by removing all redundant constraints in a certain order is also equivalent to N. □

In Algorithm 3, CollectResult() collects the non-redundant constraints obtained by each computing core and merges them into a QCN containing only non-redundant constraints. The specific implementation is to directly combine all non-redundant constraint sets $N_c^{(1)}, \ldots, N_c^{(K)}$. The specific implementation of AllocateTask() will be discussed in detail in the next subsection.

C. Allocation Algorithm Optimization

Different task allocation algorithms will affect the efficiency of the distributed algorithm. The main reason is that in the process of distributed computation, if tasks are unevenly allocated, it will cause some nodes to complete tasks much later than others, and the total calculation time will be determined by longest computation time of nodes, resulting in the so-called "long tail" phenomenon. This paper proposes the following three task allocation algorithms: first, the *natural allocation algorithm* (natural)where the tasks are evenly allocated to different computation nodes for calculation following the current order of tasks; second, the *random allocation algorithm* (random), which reorders tasks randomly and then allocates them as the natural allocation algorithm; third, the *group allocation algorithm* (group), as shown in Algorithm 4. Notably, the former two algorithms are two common allocation algorithms in distributed computation.

Algorithm 4: AllocateTask-Group

```
Input: a QCN $N = (V, C)$; a graph $G = (V, E)$; the number of computation nodes $K$.
Output: Allocation result $\{C_1, \cdots, C_K\}$.
1  for $v_i$ in $V$ do
2      $P_i \leftarrow \{N[v_i, v_j] \mid j > i, (v_i, v_j) \in E\}$;
3  end
4  $\{P_1, \ldots, P_{|V|}\} \leftarrow$ Shuffle$(\{P_1, \ldots, P_{|V|}\})$;
5  $m \leftarrow \lfloor |V|/K \rfloor$;
6  for r from 1 to K do
7      $C_r \leftarrow P_{1+m(r-1)} \cup \ldots \cup P_{m+m(r-1)}$;
8  end
9  $C_K \leftarrow C_K \cup P_{Km+1} \cup \ldots \cup P_{|V|}$;
10 return $\{C_1, \cdots, C_K\}$;
```

The algorithm first divides all tasks $\{N[v_i, v_j] \mid j > i\}$ for some v_i into one group, and then randomly distributes different groups to different computation nodes for calculation. This allocation algorithm is based on the analysis of time consumptions of the redundancy problem task.

In fact, for the group algorithm, it can be proved that it is highly probable that the runtimes of different computation nodes are close to each other. For the variable v_i, let the total calculation time be t_i, and suppose these t_i follow a certain distribution T where the mathematical expectation $E(T) = \mu$ and variance $D(T) = \sigma^2$. Now allocate the groups for $v_i \in V$ to K computation nodes $C_1, \cdots, C_K$ randomly, and the number of groups for each node is m. Suppose the total time of C_r is $U_r = t_{i_1} + \cdots + t_{i_m}$, then $E(U_1) = \cdots = E(U_K) = E(U) = m\mu$. We call $|U_r - U_s|/E(U)$ the *deviation rate*, which measures the difference between the total runtimes of two computation nodes.

Theorem 3. *When m is large enough, we have that $P(|U_r - U_s|/E(U) \leq Z_{\alpha/2}\sqrt{2}\sigma/(\sqrt{m}\mu)) = 1-\alpha$, i.e., the probability that the deviation rate does not exceed $Z_{\alpha/2}\sqrt{K}\sigma/(\sqrt{n}\mu)$ is $1 - \alpha$, where $Z_{\alpha/2}$ is the Z value of the standard normal distribution for confidence level α.*

Proof. As $U_r = t_{i_1} + \cdots + t_{i_m}$, when m is large enough, by the central limit theorem, U_r approximately follows a normal distribution with expectation of $m\mu$ and variance of $m\sigma^2$, and is independent from each other for different r. Therefore, $(U_r - U_s)/(\sqrt{2m}\sigma)$ follows a normal distribution with expectation of 0 and variance of 1. From the nature of the normal distribution, we can see that $P(|U_r - U_s|/(\sqrt{2m}\sigma) \leq Z_{\alpha/2}) = 1-\alpha$ and it can be converted to $P(|U_r - U_s| \leq Z_{\alpha/2}\sqrt{2m}\sigma) =$

$1-\alpha$, then $P(|U_r - U_s|/E(U) \leq Z_\alpha\sqrt{2m}\sigma/E(U)) = 1-\alpha$ and thus, $P(|U_r - U_s|/E(U) \leq Z_{\alpha/2}\sqrt{2}\sigma/(\sqrt{m}\mu)) = 1-\alpha$ □

For large data sets, m is generally large, and the deviation rate between different computation nodes can be approximated by the above theorem at the confidence level of α. For example, for the adm2 data set in this paper (see Table I), suppose the calculation is done by $K = 6$ nodes, then $m \approx 0.29 \times 10^6$. The sample variance $\sigma \approx 0.917 \times 10^{-2}$, and sample expectation $\mu \approx 0.121 \times 10^{-3}$. If $\alpha = 0.05$, then there is a probability of 0.95 that the deviation rate does not exceed $Z_{\alpha/2}\sqrt{2}\sigma/(\sqrt{m}\mu) \approx 0.39$. The actual situation is close to this estimate where in 28,000 groups there are only 1376 (4.91%) ones with the deviation rate exceeding 0.39. In general, the group algorithm will give a balanced allocation with a large probability, which can effectively reduce the occurrence of the "long tail" phenomenon.

Moreover, the group algorithm has a certain advantage in data reading efficiency over the random algorithm, because sequential access of the data in the memory is faster than random access. In fact, the group algorithm allocates all constraints of the same variable to the same computation node, so these constraints generally have similar memory addresses, and thus data access in this case is approximately sequential. In this way, the computation efficiency is further improved. In the next section, the above intuitive analysis will be verified through experiments.

V. Evaluations

A. Experiments Settings and Evaluation Criteria

This paper uses the public available real-world data sets used in previous studies on the redundancy problem [6], [17], including Footprint-k (F-k), StatisticalArea-k (SA-k), nuts, adm1, gadm1, gadm2, and adm2 (17 in total), as shown in Table I, where F-k, SA-k are basic networks. Since the focus of this paper is not the path consistent or partial path consistent algorithm, in order to eliminate the impact, all data sets are preprocessed with the partial path consistent algorithm. Table I shows the number of constraints after processing.

Based on the latest literature, we selected three baseline algorithms for comparative analysis, including prime [17], simple, and extended. Prime is the implementation of Algorithm 1 in this paper, and is the state of the art that is a complete algorithm (can ensure all redundant constraints are found). Simple and extended are from [5], which are incomplete algorithms. Among them, simple only checks whether there is $v_k \neq v_i, v_j$ for a $N[v_i, v_j]$ such that $N[v_i, v_k] \diamond N[v_i, v_k] \subseteq N[v_i, v_j]$, while extended needs to maintain an additional directed graph structure to assist identifying redundant constraints.

The main concern of this paper is the computation efficiency, which will be evaluated by the total time used to identify and remove all redundant constraints in a network. For the distributed algorithm proposed in this paper, we also measure the commonly used speedup ratio (the ratio of single-node computing time to multi-node computing time). The

TABLE I
Details of data sets.

	#variables	#constraints	#redundancies
SA-1	51	1275	936
SA-2	100	4950	4146
SA-3	196	19110	17395
SA-4	376	70500	66319
SA-5	659	2.1×10^5	2.1×10^5
SA-6	1562	1.219×10^6	1.200×10^6
F-1	108	5778	2542
F-2	217	23436	15618
F-3	434	93961	66106
F-4	867	3.7×10^5	2.9×10^5
F-5	1736	1.5×10^6	1.3×10^6
F-6	3470	6.0×10^6	5.7×10^6
nuts	2239	3392	1143
adm1	11766	1.5×10^6	1.49×10^6
adm2	1.7×10^6	8.8×10^6	6.9×10^6
gadm1	42753	6.6×10^6	6.5×10^6
gadm2	2.7×10^5	6.0×10^5	3.0×10^5

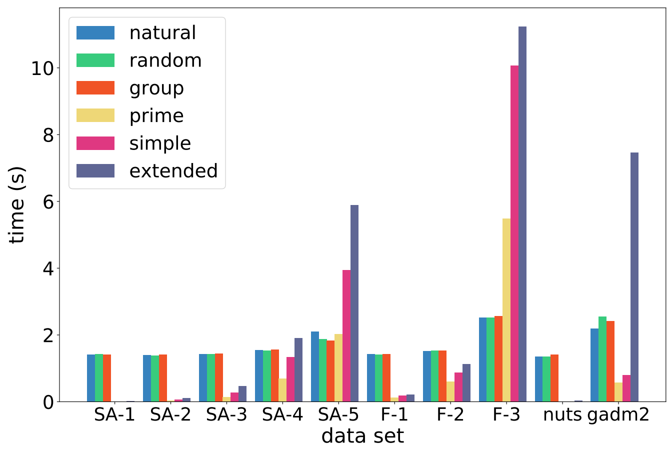

Fig. 4. Results on easier data sets.

experimental platform in this paper is Ubuntu 18.04, CPU i7-8700 3.2GHz, RAM 64GB, with Spark 3.1.1 and Python 3.6. In order to avoid the impact of network communication on the analysis of essential performance, we use multiple CPU cores on the same server as computation nodes for distributed computation. In order to analyze the influence of the number of computation nodes, we will evaluate the computation time using 1 to 6 cores with a 1 core increment.

B. Results and Analysis

To show the results clearly, this paper divides the data sets into two groups. The first group is the group with shorter computation time, called the easier data sets, and the second group is the group with longer calculation time, called the harder data sets. The results are shown in Fig. 4 and Fig. 5 (all results are repeated Average of 10 experiments).

Since the distributed algorithm has additional overhead for initialization and allocation, its advantage is not obvious for the easier data sets, and sometimes it is even slower than the baseline algorithms, which is as expected. On the other hand, for the harder data sets, in most cases distributed algorithms

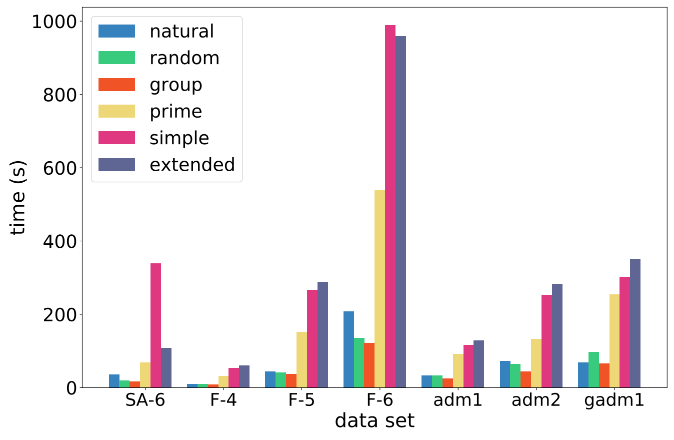

Fig. 5. Results on harder data sets.

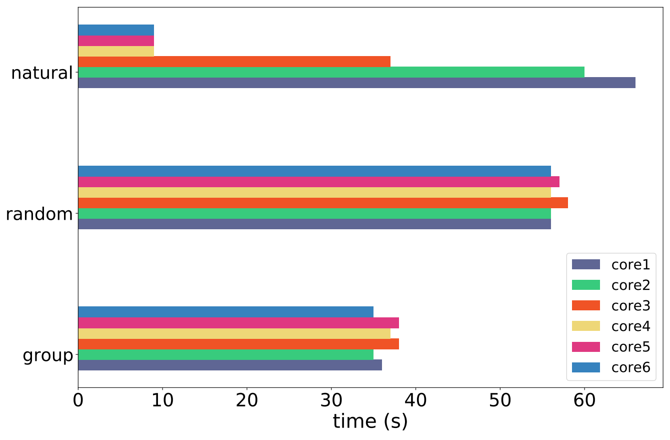

Fig. 6. Comparison of different allocation algorithms

greatly improved the computation efficiency. Specifically for adm1, gadm1, and adm2, the distributed algorithm based on group allocation (group) has speedup ratios around 4, which shows that distributed algorithms do have the potential to significantly improve the efficiency. Comparing the three different allocation algorithms, it can be seen that natural is inferior, random is slightly better, and group has the best performance. This is because natural is likely to assign time-consuming tasks to the same computing core, which results in an obvious "long tail" phenomenon. Although random has a better average performance than natural, as it allocates tasks to different cores in no order, the data access is almost random, which is less efficient. On the contrary, group allocates tasks of a variable to the same core, so the data access is sequential and is more efficient. As stated in Theorem 3, this usually makes the load of each core balanced and thus avoids an obvious "long tail" phenomenon. This can also be seen from the results in Fig. 6, where the computation time for each core is recorded for adm2 with 6 cores. Note that the total computation time for all the cores should have been the same if there is no difference in data access, as the total number of calculations are the same for any allocation algorithm. In fact, group takes about 120 seconds less than random, which shows that group indeed has the advantage in data access. In addition, natural has an obvious "long tail" phenomenon. The above results show that for the redundancy problem, the distributed algorithm with group allocation has significant advantages.

Figs. 7 and 8 are the results of the distributed algorithms based on the three allocation algorithms with varying number of cores on the easier data set SA-5 and the harder data set adm2. On the harder data set, the speedup ratio usually increases as the number of cores increases, but the rate of change becomes smaller gradually. This is normal, because increasing the number of cores causes additional overhead. On the easier data set, since the overhead of distributed algorithm accounts for a large proportion, increasing the number of cores does not have significant improvement on the computation time and can sometimes even increase the time instead. This shows that the distributed algorithm proposed in this paper is more appropriate for more complex large-scale data sets, and it is necessary to consider the balance between economy and efficiency.

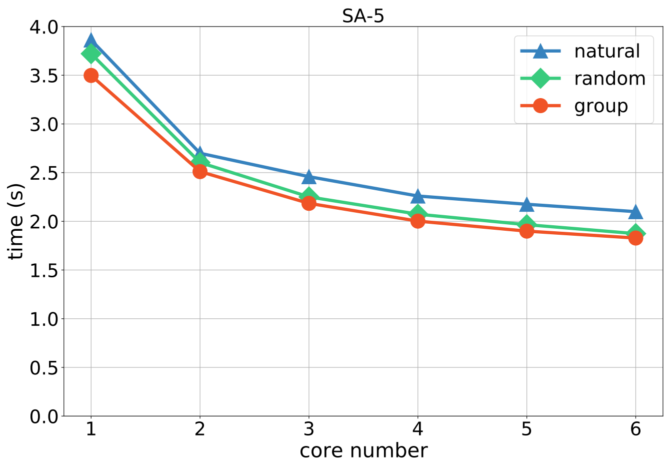

Fig. 7. Results on the easier data set SA-5 by varying the number of computation cores.

In summary, the distributed approach proposed in this paper can significantly improve the efficiency of the redundancy algorithm, and thus enhance the ability to process more complex large-scale data, which is a significant improvement of the existing algorithms.

VI. Summary and Outlook

This paper proposed a novel distributed model that can significantly improve the efficiency of identifying and removing redundant constraints in large-scale data sets. By analysing the properties of the distributed subalgebra, we proved the feasibility of the distributed model. Based on the characteristics of qualitative relations in data sets, we proposed optimizations of the model and demonstrated their effectiveness. Finally, the performance of the proposed model

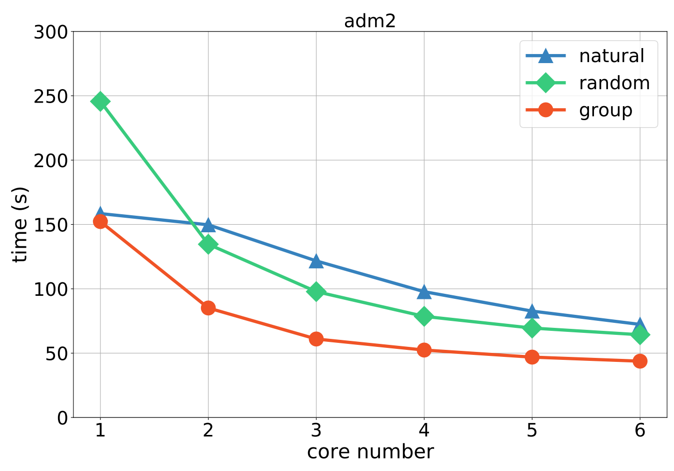

Fig. 8. Results on the harder data set adm2 by varying the number of computation cores.

and optimizations is analysed through extensive experiments, comparing baseline algorithms and different optimizations on multiple real-world data sets. The experimental results verified that the distributed model performs well on the redundancy problem for large-scale data sets. Since distributed algorithms enable data privacy between different computation nodes, in the future we will study the mechanism of the proposed algorithm to protect privacy and security for qualitative spatio-temporal data.

Acknowledgment

We thank anonymous reviewers for their helpful comments and suggestions.

References

[1] M. Al-Omari, P. Duckworth, N. Bore, M. Hawasly, D. C. Hogg, and A. G. Cohn, “Grounding of human environments and activities for autonomous robots,” in *IJCAI*, 2017, pp. 1395–1402.

[2] D. A. Randell, G. Landini, and A. Galton, “Discrete mereotopology for spatial reasoning in automated histological image analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 568–581, 2013.

[3] J. Suchan, M. Bhatt, P. A. Walega, and C. P. L. Schultz, “Visual explanation by high-level abduction: On answer-set programming driven reasoning about moving objects,” in *AAAI*, 2018, pp. 1965–1972.

[4] N. Krishnaswamy, S. Friedman, and J. Pustejovsky, “Combining deep learning and qualitative spatial reasoning to learn complex structures from sparse examples with noise,” in *AAAI*, 2019, pp. 2911–2918.

[5] J. O. Wallgrün, “Exploiting qualitative spatial reasoning for topological adjustment of spatial data,” in *ACM-GIS*, 2012, pp. 229–238.

[6] J. F. Condotta, S. Kaci, P. Marquis, and N. Schwind, “Majority merging: From Boolean spaces to affine spaces,” in *ECAI*, 2010, pp. 627–632.

[7] S. Du, L. Luo, W. Zhao, and Z. Guo, “Research progress in multi-scale spatial relations (Chinese),” *Journal of Geo-Information Science*, vol. 17, no. 2, pp. 135–146, 2015.

[8] S. Li and M. Ying, “Region Connection Calculus: Its models and composition table,” *Artificial Intelligence*, vol. 145, no. 1-2, pp. 121–146, 2003.

[9] Z. Long, S. Schockaert, and S. Li, “Encoding large RCC8 scenarios using rectangular pseudo-solutions,” in *KR*, 2016, pp. 463–472.

[10] Z. Long, H. Meng, T. Li, and S. Li, “Compact geometric representation of qualitative directional knowledge,” *Knowledge-Based Systems*, vol. 195, p. 105616, 2020.

[11] J. Ouyang, “Research on some problems in spatio-temporal reasoning (Chinese),” Ph.D. dissertation, Jilin University, 2005.

[12] J. Chen, D. Liu, H. Jia, and C. Zhang, “Integrative reasoning with topological, directional and size information based on MBR (Chinese),” *Journal Of Computer Research and Development*, vol. 47, no. 3, pp. 426–426, 2010.

[13] M. Sioutis, A. Paparrizou, and J. Condotta, “Collective singleton-based consistency for qualitative constraint networks: Theory and practice,” *Theoretical Computer Science*, vol. 797, pp. 17–41, 2019.

[14] M. Sioutis, Z. Long, and T. Janhunen, “On robustness in qualitative constraint networks,” in *IJCAI*, 2020, pp. 1813–1819.

[15] S. Wang and S. Zhang, “Qualitative spatial reasoning and its application (Chinese),” *Communications of CCF*, vol. 11, no. 11, pp. 22–26, 2015.

[16] S. Wang, J. Shao, N. Li, C. Liu, W. Li, P. Chen, and B. Liu, “Localization and labeling of intervertebral disc based on deep learning and spatial relation reasoning (Chinese),” *Journal of Jilin University*, vol. 50, no. 209, pp. 326–331, 2020.

[17] A. Ginsberg, “Knowledge-base reduction: A new approach to checking knowledge bases for inconsistency and redundancy,” in *AAAI*, 1988, pp. 585–589.

[18] J. G. Schmolze and W. Snyder, “Detecting redundancy among production rules using term rewrite semantics,” *Knowledge-Based Systems*, vol. 12, no. 1–2, pp. 3–11, 1999.

[19] G. Gottlob and C. G. Fermüller, “Removing redundancy from a clause,” *Artificial Intelligence*, vol. 61, no. 2, pp. 263–289, 1993.

[20] P. Liberatore, “Redundancy in logic I: CNF propositional formulae,” *Artificial Intelligence*, vol. 163, no. 2, pp. 203–232, 2005.

[21] M. J. Egenhofer and J. Sharma, “Assessing the consistency of complete and incomplete topological information,” *Journal of Geographical Systems*, vol. 1, no. 1, pp. 47–68, 1993.

[22] M. A. Rodríguez, M. J. Egenhofer, and A. D. Blaser, “Query preprocessing of topological constraints: Comparing a composition-based with neighborhood-based approach,” in *SSTD*, 2007, pp. 362–379.

[23] S. Li, Z. Long, W. Liu, M. Duckham, and A. Both, “On redundant topological constraints,” *Artificial Intelligence*, vol. 225, pp. 51–76, 2015.

[24] Z. Long and S. Li, “On distributive subalgebras of qualitative spatial and temporal calculi,” in *COSIT*, 2015, pp. 354–374.

[25] M. Sioutis, S. Li, and J.-F. Condotta, “Efficiently characterizing non-redundant constraints in large real world qualitative spatial networks,” in *IJCAI*, 2015, pp. 3229–3235.

[26] M. Mantle, S. Batsakis, and G. Antoniou, “Large scale reasoning using Allen’s Interval Algebra,” in *MICAI*, 2017, pp. 29–41.

[27] S. Nam and I. Kim, “Mrqusar: A web-scale distributed spatial reasoner using MapReduce,” in *BigComp*, 2017, pp. 296–303.

[28] J. Kim and I. Kim, “Scalable distributed temporal reasoning,” in *CSA/CUTE*, vol. 474, 2017, pp. 829–835.

[29] M. Mantle, S. Batsakis, and G. Antoniou, “Large scale distributed spatio-temporal reasoning using real-world knowledge graphs,” *Knowledge-Based Systems*, vol. 163, pp. 214–226, 2019.

[30] D. A. Randell, Z. Cui, and A. G. Cohn, “A spatial logic based on regions and connection,” in *KR*, 1992, pp. 165–176.

[31] D. A. Randell, A. Galton, S. Fouad, H. Mehanna, and G. Landini, “Mereotopological correction of segmentation errors in histological imaging,” *Journal of Imaging*, vol. 3, no. 4, p. 63, 2017.

[32] A. Chmeiss and J.-F. Condotta, “Consistency of triangulated temporal qualitative constraint networks,” in *ICTAI*, 2011, pp. 799–802.

[33] Z. Long and S. Li, “On distributive subalgebras of qualitative spatial and temporal calculi,” in *COSIT*, 2015, pp. 354–374.