

RESEARCH ARTICLE

An Artificial Intelligence Technology Based Algorithm for Solving Mechanics Problems

JIARONG ZHANG¹, JINSHA YUAN^{1,2}, AND JIANING XU²¹Department of Electronic and Communication Engineering, North China Electric Power University, Baoding 071000, China²Power Dispatching Control Center, State Grid Fuxin Power Supply Company, Fuxin 123000, China

Corresponding author: Jiarong Zhang (zjraaaa@126.com)

This work was supported in part by the Fundamental Research Business Expenses of Central Universities (CN) under Grant 2020JG006.

ABSTRACT As an indispensable technology of intelligent education, intelligent tutorial algorithms for solving mathematical or physical problems have attracted much attention in recent years. Nevertheless, since solving mechanics problems requires complex force analysis and motion analysis, current researches are mainly focus on solving geometry proof problems and direct circuit problems. There are some inherent challenges on developing such algorithms, including the low intelligence, mobility and interpretability of the comprehension algorithm. Therefore, this article develops a novel algorithm for solving mechanics problems. First, we propose a comprehension model for mechanics problems and convert problem understanding into relation extraction. Furthermore, a novel neural model combining pretrained model BERT and graph attention network (GAT) is proposed to extract the direct conditions of input mechanics problems. Second, a hidden information mining method is proposed for supplementing the conditions of the input problem. Third, a predicate logic based algorithm is proposed for force analysis. Finally, a solving algorithm is presented for choosing equations to acquire the solutions. Solving experiments and sensitivity analysis are provided to demonstrate the effectiveness of the proposed algorithm.


INDEX TERMS Intelligent tutoring, mechanics problems, graph attention network, predicate logic, knowledge graph.

I. INTRODUCTION

As an interdisciplinary subject of pedagogy and artificial intelligence (AI), educational information technology (EIT) plays an increasingly important role around the world. Thus, intelligent tutorial algorithms for solving different subjects, which convert input problems into readable solutions, have attracted much attention from both industry and academia [1], [2], [3], [4], [5]. However, current researches on intelligent tutorial algorithms are mainly concentrated on mathematical word problems, geometry proof problems and direct circuit problems. There are rare study on mechanics problems. Moreover, as most algorithms cannot automatically understand problem, solving problems or explain the generated answers in detail, most existed tutorial functions are not in high intelligence. Finally, existing algorithms can only solve a single problem, and cannot integrate different types

of problems to solve them together. Therefore, constructing an intelligent tutorial algorithms with high capabilities for solving mechanics problems is of great significance and can promote the development of EIT.

Current researches on intelligent tutorial algorithms can be divided into four categories. The first category is rule based method, which are constructed by rules or templates. This type methods are adopted in the early stage and studies were reported in [6], [7], [8], and [9]. The second category is statistics based method, which are constructed by traditional classifier. For example, Mitra *et al.* [10] presented three pre-defined template for solving math problems. Roy *et al.* [11] proposed three classifiers to detect the attributes of word problems for solving one operator arithmetic problems. He *et al.* [12] employ a S^2 [13] model to acquire relations of circuit problem for the solution. Other researches include [10] and [14]. The third category is tree based method. These methods solving a problem by transforming the arithmetic expression into formula tree. Roy *et al.* [15] proposed the first tree based

The associate editor coordinating the review of this manuscript and approving it for publication was Bo Pu .

algorithm for solving arithmetic word problem. The algorithm regards solution as tree construction and turns it into a classification task. A brute force search method based on linear programming presented by Koncel-Kedziorski *et al.* [16] for parsing algebraic word problems into equations. The final category is deep learning (DL) based method, which using deep neural network (DNN) to train an end-to-end model. Wang *et al.* [17] proposed deep neural solver (DNS) for math word problems by constructing a sequence-to-sequence (Seq2Seq) model based on recurrent neural network (RNN). Wang *et al.* [18] presented an ensemble model math equation normalization (Math-EN) method on the basis of DNS. Xie *et al.* [19] conducted an algorithm which generate expression tree in a goal-driven manner by using a tree structured (GTS) neural model. They generate expression trees step by step by decomposing the encoded target into sub-goals and a two-layer gated-feedforward networks is designed for implementing each step of goal decomposition. Zhang *et al.* [20] combined a graph-based encoder and a tree-based decoder to propose a novel deep learning framework GTL for improving performance on mathematical problems. Zhang *et al.* [21] combined deep learning with predicate logic to propose an intelligent tutorial algorithm DLR for solving kinematics problems. They employ BERT [22] to extract information of the input question and use predicate logic to interpret the output solutions.

These algorithms still has many limitations and shortcomings. First, the rule based method and the statistics based method require manual formulation of rules and templates, which resulting in low generalization capabilities. Second, a deficiency of tree-based methods is the search space of the tree grow exponentially while the increasing of the quantities. Third, for deep learning-based algorithms, it is impossible to achieve readable understanding, analysis and solutions through a unified framework. Finally, most researches are focus on mathematics problems and direct circuit problems in elementary or junior high schools. The research on mechanics problems falls behind these fields, and existing mechanics tutorial algorithms are not competent to force analysis. Therefore, it is high time to construct an intelligent algorithms for solving mechanics problems.

There are three challenges in developing such an algorithm. First, mechanics is composed of kinematics and dynamics, the main task of dynamics is force analysis while motion analysis for kinematics. How to accomplish the kinematics analysis and force analysis through a unified framework is the basic challenge. Second, different from mathematical word problems, geometry proof problems and direct circuit problems, mechanics studies specific objects in life, such as cars and airplanes. Thus, solving a mechanics problem requires not only model knowledge like mathematical problems, but also general knowledge. For example, the description of problem 1 (P1) is “A car starts to move in a straight line from a standstill under constant traction, and passes 8m in 4s. After that, the engine is turned off and the car moves for 2s to stop. The mass of the car is known to be $m = 2 * 10^3 kg$,

ask: (1) the speed of the car when the engine is turned off; (2) the traction of the car.”. The friction of the car from the ground is not given in the text and people solve P1 by using the common sense that cars usually drive on the ground and the friction factor between the car and the ground is not equal to zero. Moreover, information such as the next state of the car after braking is stationary, and the final speed of the current state is equal to the initial speed of the next state, etc. may not given in the description. This article refers to this situation as hidden conditions missing. Third, developing an automatic force analysis algorithms is the final challenge for solving mechanics problems.

To overcome these obstacles, we employ BERT [22], graph attention network (GAT) [23], reasoning and knowledge graph (KG) [24] to construct an intelligent algorithm for solving mechanics problems. For the first challenge, we propose a sharing framework based on dual process theory (DPT) [25] in cognitive science. The sharing framework integrates the kinematics problems solving and the dynamics problems solving. And a BERT and graph attention neural network (GAT) based model is presented for acquiring the direct entities of the input problem. For the second challenge, this article constructs a general knowledge graph and employs the idea of default logic [26] to mine hidden conditions. For the final challenge, we propose a predicate logic based force analysis method and a equation selection algorithm to reason the answer.

This work develops an intelligent algorithm for solving mechanics problems. The specific contributions are listed as follows.

- 1) We develop an intelligent tutorial algorithm for solving mechanics problems, filling the gap in intelligent education on physical domain.
- 2) We propose an comprehension model for mechanics problems to convert problem understanding into relation extraction, and propose a neural model based on BERT and GAT for acquiring given conditions of input problem.
- 3) We propose a logic-based hidden information mining algorithm to complement the given conditions of the input mechanics problem.
- 4) We propose a predicate logic based method for force analysis and a template based method for transform predicate into equations. And an equation generation algorithm and an equation selection algorithm are proposed to reason the answer.

The rest of this article is organized as follows. Section II presents the modeling framework, including deep learning based natural language understanding, knowledge graph based comprehension reasoning and predicate logic based solution reasoning. In section III, extensive experiments are presented, including a performance evaluation, a comparison experiment and a sensitivity analysis. Finally, Section IV provides a discussion and Section V concludes this article.

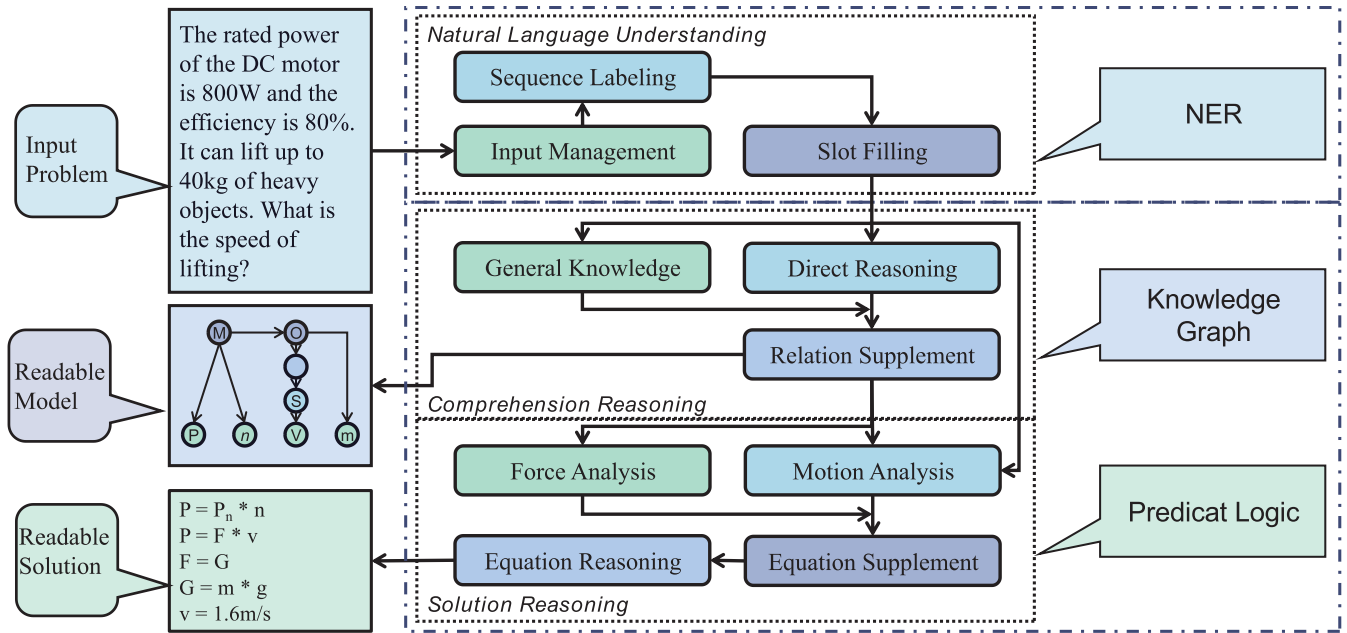


FIGURE 1. Overview of the proposed algorithm. Natural Language Understanding aims to convert natural language into named entities. Comprehension reasoning aims to transform entities sequences into physical models represented by predicate logic by reasoning. Solution Reasoning aims to acquire the answer.

II. MODELING FRAMEWORK OF THE PROPOSED ALGORITHM

We aim to acquire the readable analysis and solution based on the input mechanics problem. As the given conditions are provided by natural language, we first need to convert it to a structured physical model and then solve the input problem accurately. The overview of the proposed algorithm is provided in FIGURE 1. There are three main steps in the algorithm. First, the input natural language are proposed by a natural language understanding procedure incorporating input management, sequence labeling and slot filling. Second, a comprehension reasoning is built by using predicate logic based on the results from the first step. Third, a solution reasoning procedure is applied to solve the problem when the predicate representation of the physical model is obtained. The details are illustrated in the following sections.

A. DEEP LEARNING BASED NATURAL LANGUAGE UNDERSTANDING

People usually transform the problem in natural language into a structured physical model to understand mechanics problems. Thus, in this section, we first propose a comprehension model for mechanics problems. Then, we structurally represent the proposed model in the form of relational triples. In this way, the mapping from natural language to comprehension model of input problem is transformed into a relation extraction task. However, there are overlapping entities in the input problem, and existing methods cannot achieve satisfactory results. Moreover, the hidden condition of the input problem are usually attributes of entities in the text. Therefore, we use a novel tagging scheme based on the proposed comprehension model and propose a BERT-GAT

based algorithm to recognize the entities first. And then reasoning the relations according the tagging schema.

1) COMPREHENSION MODEL OF MECHANICS PROBLEMS

As mentioned above, we understand the input problem by mapping the problem to a comprehension model. To clarify this model, we use an example for demonstration, as shown in FIGURE 2. The description of problem 2 (P2) is “The car changes from a static state to a uniformly accelerated linear motion with an acceleration of $a_1 = 0.5m/s^2$, and changes to a uniform linear motion after 10s. When the uniform motion lasts for 10s, the car brakes suddenly because it encounters an obstacle. The acceleration of the brake is known. $a_2 = -2m/s^2$, find: (1) The speed of the car at a constant speed; (2) The displacement of the car within 36s.”. It can be concluded from the description that P2 has only one object and four states, where UALM denotes uniformly accelerated linear motion, ULM denotes uniform linear motion.

Two observations can be made from FIGURE 2: First, the tree-like comprehension model can be divided into four levels: object, ORstate, state and condition. The first level aims to demonstrate the research object of the input problem. The second level is used to clarify the overall movement of the object. The third level is composed of several specific motion state which detail the motion of the object. The last level demonstrates the given conditions of the input problem, such as velocity, time, force, etc. Second, each entity in the figure is connected to other entities by one or more directed lines. This is the same representation as relational triples. So we use a set of relational triples to represent the proposed comprehension model. In turn, the understanding the input problem is transformed into a relation extraction task. Denoting $D(P)$

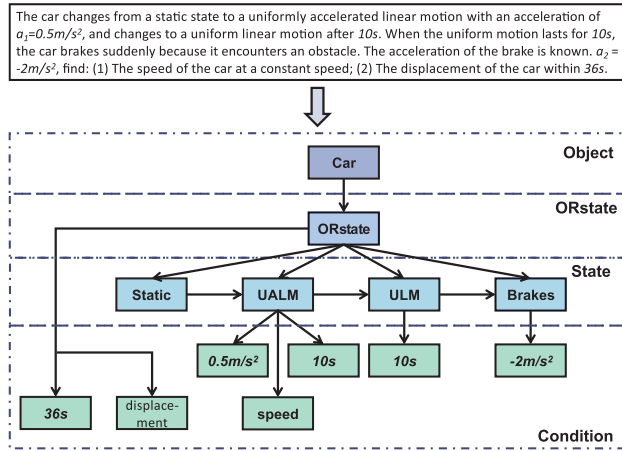


FIGURE 2. The proposed comprehension model, which contains four levels: object, ORstate, state, condition. Each rectangle represents an entity in the problem. UALM indicates uniformly accelerated linear motion, ULM indicates uniform linear motion.

as the discription of input problem, the transformation can be written as

$$D(P) \rightarrow T \tag{1}$$

where $T = \{r_1, r_2, \dots, r_m\}$ denotes the triples set of comprehension model, m is the number of triple. Each elements in T called a triple, which can be expressed as

$$r_i = (e_{i1}, p_i, e_{i2}) \tag{2}$$

where r_i denotes the i -th element of T , e_{i1} is an entity called the subject of r_i , p is a relation called predicate of r_i , e_{i2} is an entity called the object of r_i .

2) A NOVEL TAGGING SCHEMA FOR RELATION EXTRACTION

As mentioned, understanding is to extract triples in input problem. As the needs of solving mechanics problems, we propose a novel tagging schema and using named entity recognition (NER) and reasoning for extracting triples. The reasons are as follows:

First, a description of a mechanics problem is usually a long text. The text usually contains many (greater than 5) relation triples. Existing end-to-end algorithms cannot handle such complex situations.

Second, as provided in FIGURE 2, there are two “10s” in the model. But they have different meanings, and we call this situation as entity overlap. These overlapping entities affect the performance of relation extraction. Moreover, the proposed model is a tree model, which an entity and different entities form multiple triples, which is a difficulty in relation extraction. For mechanics problems, entities in the text contain various information, direct end-to-end extraction may ignore these important information.

Third, using conditions provided in FIGURE 2 to calculate will lead to erroneous results. The reason is the next state of brake is static according the general knowledge, but there is no caption in the input text. Therefore, the complete state level of P2 should add a static after brake. Moreover, some

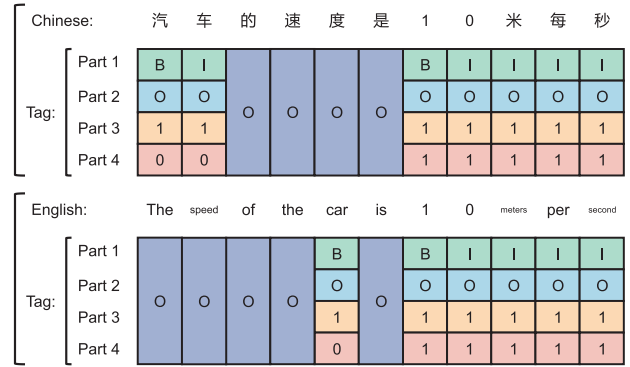


FIGURE 3. The proposed tagging schema, which contain four parts. Part 1 is the boundary label. Part 2 is the category label. Part 3 is object label. Part 4 is state label.

critical conditions are not given in the input text, such as the velocity of static is $0m/s$, the result force of static is $0N$, the final velocity of UALM is equal to the initial velocity of ULM, etc. These hidden conditions are requisite when listing equations. All these conditions are associated with recognized entities in FIGURE 2. So those end-to-end algorithms don’t work for this scenario.

Therefore, we employ the model based on the pipeline structure, which first use a neural model to identify entities, and then determine the relationship between entities through another neural model. To eliminate error propagation, we replace the second neural model with label information based reasoning.

Although reasoning outperforms neural networks when knowledge is complete, the premise is that sufficient information is available. Traditional tagging schemes only contain boundary information and category information, which cannot infer correct relation triples. For example, we cannot infer the relation between $0.5m/s^2$ and UAML in P2 from these two information. By analyzing the proposed understanding model, we find that for any entity, when we know which object and which state is its parent node, we can infer its relation with other entities. So we establish a novel tagging schema by adding object information and state information to the traditional schema. An example is provided in FIGURE 3 to clarify the proposed tagging schema. There four part for each entity. First, part 1 is the boundary label to illustrate the begin (B), inside (I) or out (O) of an entity. Second, part 2 is the category label to represent the category of an entity, object (O), velocity (V) or another type. Third, part 3 is the object label to indicate which object is the parent node of current entity. Finally, part 4 is state label which aims to determine the state information of current entity.

3) NAMED ENTITY RECOGNITION USING BERT-GAT

Different with English sentences, lexicons in Chinese sentences are difficult to demarcate. However, lexical information is beneficial for Chinese named entity recognition. To efficiently recognize named entities from Chinese text using lexical information, different models have been

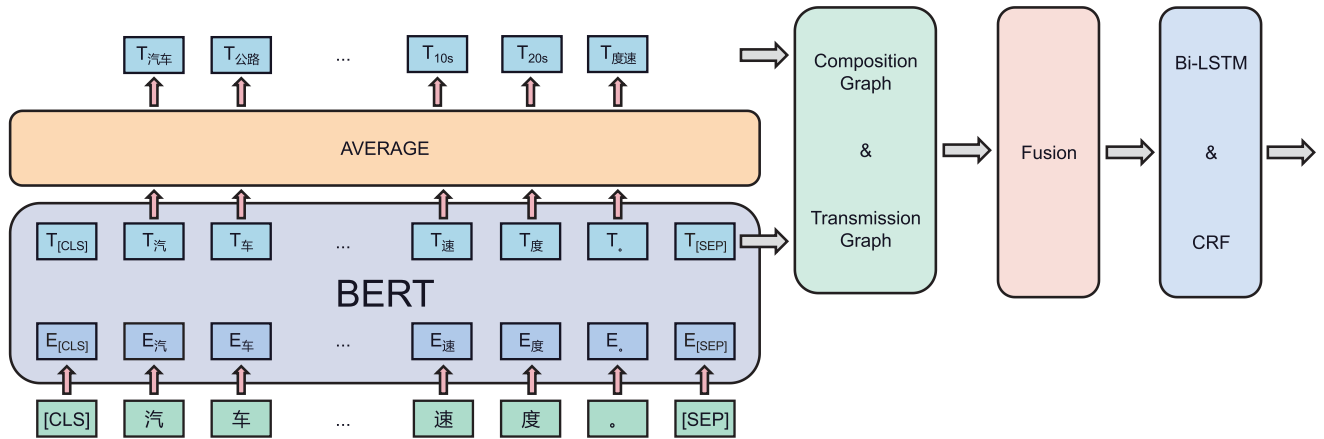


FIGURE 4. The architecture of the proposed neural model, containing BERT encoder, graph based encoder and sequence learning based decoder.

proposed in the literature. Zhang *et al.* [27] proposed Lattice LSTM model to reduce word segmentation errors. However, this method only uses the boundary information of lexicons and ignores the composition information of the vocabulary. With a similar purpose, graph neural networks [28] are also widely used in Chinese named entity recognition [29]. Although these methods are easy to implement, they do not fully utilize the lexical information, including composition information and transmission information.

In this work, we adopt BERT [22] and GAT [23] for recognizing entities in Chinese text, which take advantage of large-scale pretraining model and graph model. The model architecture is provided in FIGURE 4. There are main three steps in the method. First, the raw sentence is processed by BERT model to acquire the node features, including character features and lexicon features. Second, two graph model (composition graph and transmission graph) and a fusion model are proposed to obtain final embedding. Third, a Bi-LSTM & CRF model is applied to achieve tagged sequence based on the results from the second step. The detailed procedure is provided as follows.

BERT is a pretraining model based on transformer encoder [30]. Different with traditional convolutional neural networks (CNNs) and recurrent neural networks (RNNs), transformer encoder employ a multi-head self attention mechanism for long-term dependencies. The propagation formulas are written as

$$\begin{cases} Multihead(Q, K, V) = concat(head_i)W^O \\ head_i = Attention(Q_iW_i^Q, K_iW_i^K, V_iW_i^V) \end{cases} \quad (3)$$

where $Q, K,$ and V denote the query vector, key vector and value vector respectively; $head_i$ denotes the single-head self-attention mechanism layer; W^O denotes the weight matrix; $W_i^Q, W_i^K,$ and W_i^V denote the projection matrices. Moreover, the calculation of attention uses the scaled dot-product format, which can be written as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

where d_k denotes the dimension of the input vector.

BERT for Chinese can only output the vector of input characters, where as, lexical vectors are essential information for processing Chinese sentences. To acquire the lexicon features, we propose an averaging method. The propagation formula is

$$T_{w_i} = \frac{1}{N} \sum_{j=1}^N T_{c_j} \quad (5)$$

where T_{w_i} denotes the node feature of the i -th word w_i , T_{c_j} denotes the node feature of character c_j . Moreover, c_j denotes the j -th character makes up w_i , and there w_i is composed of N characters.

To fully use the contextual information and lexical information, two character and lexicon based graph are proposed for modeling a Chinese sentence. The first graph is transmission graph which is constructed for contextual information and the second is composition graph which is constructed for lexical information. As the characters and words are fixed for the same sentence, the two graphs share the same nodes set, but the connection patterns are different.

There are two types of nodes in a sentence: character node and lexicon node. Denoting S as the input sentence, then each character in S corresponds to a character node and each word in S that matches the predefined vocabulary corresponds to a word node. In this work, c represents a character node and w represent a lexicon node. Assuming that sentence 1 (S_1) has nine characters ($c_1 - c_9$) and six lexicons ($w_1 - w_6$) matching the predefined vocabulary, in which $c_1 - c_2$ constitute w_1 , $c_1 - c_3$ constitute w_2 , $c_4 - c_5$ constitute w_3 , $c_4 - c_9$ constitute w_4 , $c_6 - c_7$ constitute w_5 , $c_8 - c_9$ constitute w_6 . The details of constructing the two graphs of S_1 are as follows.

The contextual information can be divided into character contextual information and word contextual information. Character contextual information is the sequential information of characters, while word contextual information describe the word-to-word information. As provided in FIGURE 5, if i and j are two adjacent character nodes, and j is the following node of i , then (i, j) of the transmission graph adjacency matrix A_T will be assigned 1. Furthermore, if k is

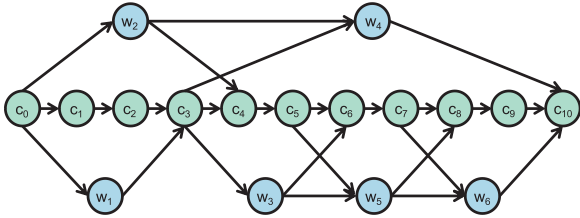


FIGURE 5. The transmission graph of S1, where *c* denotes character node and *w* denotes lexicon node.

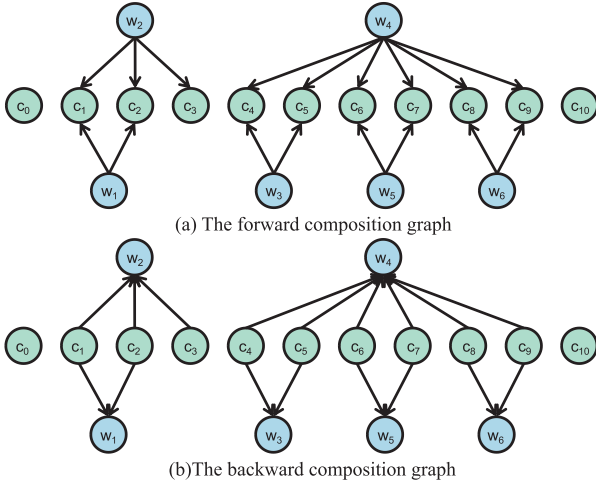


FIGURE 6. The composition graph of S1, where *c* denotes character node and *w* denotes lexicon node.

a word node and node *i* is the nearest preceding node of *k*, node *j* is the nearest following node of *k*, then (i, k) and (k, j) of A_T are assigned 1.

The composition graph describe the begin, inside and end relation between characters and lexicons. Compared with lattice LSTM [27], the composition graph can better delineate the inside information between characters and lexicons, which is crucial to extract named entities. As provided in FIGURE 6, if character node *i* is the begin, middle or end of word node *j*, then (j, i) of the forward composition graph adjacent matrix A_{FI} will be assigned 1. If character node *i* is the begin, middle or end of word node *j*, then (i, j) of the backward composition graph adjacent matrix A_{BI} will be assigned 1. Finally, perform an OR operation on each position element of A_{FI} and A_{BI} to get the composition graph adjacent matrix A_I .

In this work, we employ GAT [23] to integrate the proposed two graphs. GAT is a representative work of spatial graph neural networks. Let $F_j = \{f_{j,1}^N, \dots, f_{j,k}^N\}$ be the input node feature of the *j*-th layer GAT, where $f_{j,i}^N$ is an *N*-dimensional vector denoting the *i*-th component of F_j . Take $F_{j+1} = \{f_{j+1,1}^{N'}, \dots, f_{j+1,k}^{N'}\}$ as the output of this layer, where $f_{j+1,i}^{N'}$ is an N' -dimensional vector denoting the *i*-th component of F_{j+1} . The feature update formulas can be written as

$$\alpha_{ij}^k = \frac{\exp(\text{LeakyRelu}(a^T [W^k f_{j,i}^N \| W^K f_{j,i}^N]))}{\sum_{q \in \Omega_i} \exp(\text{LeakyRelu}(a^T [W^k f_{j,i}^N \| W^K f_{j,q}^N]))} \quad (6)$$

$$f_{j+1,i}^{N'} = \sigma \left(\sum_{k=1}^K \sum_{j \in \Omega_i} \alpha_{ij}^k W^k f_{j,j}^N \right) \quad (7)$$

where α_{ij}^k denotes attention coefficients; $W^k \in R^{F' \times F}$, $a \in R^{2F'}$ are trainable parameters; Ω_i is the neighborhood node set of node *i*; *K* is the number of attention heads. Additionally, the output of the final GAT layer is

$$f_{f,i}^{Nf} = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \Omega_i} \alpha_{ij}^k W^k f_{f-1,j}^{Nf-1} \right) \quad (8)$$

As mentioned above, the input character features and lexicon features of a sentence contains *n* characters and *m* lexicons are expressed as

$$v_I = \{T_{c_1}, \dots, T_{c_n}, T_{w_1}, \dots, T_{w_m}\} \quad (9)$$

where v_I denotes the out of BERT. To make the calculation clear, we rewrite the above formula as

$$F = \{f_0^N, f_1^N, f_2^N, \dots, f_{n+m+1}^N\} \quad (10)$$

where f_0^N and f_{n+m+2}^N denotes the start feature and end feature of input sentence, f_i^N denotes the *i*-th node feature. Denoting GAT as the function of a GAT layer, the updated node features are

$$U = \text{GAT}(F, A) = \{f_0^U, f_1^U, \dots, f_{n+m+1}^U\} \quad (11)$$

where U denotes the updated node features, $A \in R^{N \times U}$ denotes the adjacent matrix of the input nodes, f_i^U denotes the *i*-th node feature is a vector with *U* dimension. As there are two graph models are needed, we employ two independent GATs to capture the node feature and the details are written as

$$U_T = \text{GAT}(F, A_T) U_C = \text{GAT}(F, A_C) \quad (12)$$

where U_T denotes the updated node feature of transmission graph, U_C denotes the updated node feature of composition graph, A_T denotes the transmission graph adjacency matrix, A_C denotes the composition graph adjacency matrix. Finally, only the first *n* columns of the node feature are retained as the final node feature

$$E_T = U_T[:, 0:n] E_C = U_C[:, 0:n] \quad (13)$$

where E_T denotes the final node features of transmission graph, E_C denotes the final node feature of composition graph. The Fusion Layer aims to integrate the transmission node feature and the composition node feature. The propagation formulas of the fusion layer in this article is

$$I = W_T E_T \| W_C E_C \quad (14)$$

where I denotes the output of Fusion Layer, W_T denotes a trainable matrix for E_T , W_C denotes a trainable matrix for E_C .

To capture the sequence information, we applied a Bi-LSTM [31] to the results of fusion layer. Denoting LSTM as

the function as LSTM [32]. Then the propagation formulas can be expressed as

$$\begin{cases} \vec{h}_i = \overrightarrow{LSTM}(x_i, \vec{h}_{i-1}) \\ \overleftarrow{h}_i = \overleftarrow{LSTM}(x_i, \overleftarrow{h}_{i+1}) \\ \hat{h}_i = [\vec{h}_i, \overleftarrow{h}_i] \end{cases} \quad (15)$$

where \vec{h}_i and \overleftarrow{h}_i denote the positive LSTM hidden state and the reverse LSTM hidden state of the i -th vector x_i respectively, \overrightarrow{LSTM} denotes the positive LSTM function, \overleftarrow{h}_i denotes the reverse LSTM function, \hat{h}_i denotes the final hidden state of x_i .

CRF [33] has transfer characteristics, which can consider the order of output tags. Therefore, a CRF is chosen for processing the output of Bi-LSTM. After decoding the output of Bi-LSTM layer by using Viterbi algorithm [34], the tagged sequence is acquired.

B. KNOWLEDGE GRAPH BASED COMPREHENSION REASONING

Comprehension reasoning aims to acquire the complete relation set of comprehension model and output a readable model based on the results of natural language understanding. We divide relations into two categories: direct relations, which can be obtained directly in the text, and hidden relations, which require common sense knowledge. The details of comprehension reasoning are provided as follows.

1) DIRECT RELATION REASONING

Predicate logic, which consists of variables, constants, predicates, etc., has been widely used in knowledge based systems for its naturalness. In this work, we use uppercase letters indicate constants and lowercase letters indicate variables. It is well known that predicate logic can modeling modeling complex relationships. For example, denoting $Human(x)$ as x is human, $Male(x)$ as x is male. By common sense, we know that if x is human and x is male, we can conclude that x is a man. Let $Man(x)$ means x is a man, this rule can be expressed as

$$Human(x) \wedge Male(x) \rightarrow Man(x) \quad (16)$$

where $Human(x)$ and $Male(x)$ called the body and $Man(x)$ called head.

As the proposed comprehension model is a tree model, the triple set acquiring of comprehension model can be transformed into a process of finding leaf nodes based on root nodes. Thus, we sequentially reason the direct relationship in the following order.

- 1) Relations between objects.
- 2) Relations between objects and overall state (ORstate).
- 3) Relations between overall state (ORstate) and states.
- 4) Relations between states and conditions or between objects and conditions.

Relations between objects can be divided into three categories: the relation between research objects, the relation between research object and reference object and the relation

between power source (electric motor or engine) and research object. As objects are the root node of the comprehension model, these relations can be acquired by using category label and object label. The rule for reasoning the relation between research objects is as follows

$$\begin{aligned} &Object(e_1) \wedge Object(e_2) \wedge RAction(e_3) \\ &\wedge OBJ_LAB(e_1, o_1) \wedge OBJ_LAB(e_2, o_2) \\ &\wedge OBJ_LAB(e_3, o_3) \wedge STA_LAB(e_3, s_3) \\ &\wedge EQ_TO(o_1, o_3) \wedge EQ_TO(o_2, s_3) \\ &\rightarrow RACT_S(e_3, e_1) \wedge RACT_O(e_3, e_2) \end{aligned} \quad (17)$$

where $Object(x)$ denotes the category information of x research object, $RAction(x)$ denotes the category information of x is action between research objects (catch up, opposite move, etc.), $OBJ_LAB(x, y)$ denotes the object information of x is y , $STA_LAB(x, y)$ denote the state information of x is y , $EQ_TO(x, y)$ denotes x is equal to y , $RACT_S(x, y)$ denotes the subject of x is y and the category information of x is action between research objects, $RACT_O(x, y)$ denotes the object of x is y and the category information of x is action between research objects.

The rule for reasoning the relation between research objects and reference object is as follows

$$\begin{aligned} &Object(e_1) \wedge RObject(e_2) \wedge Position(e_3) \\ &\wedge OBJ_LAB(e_1, o_1) \wedge OBJ_LAB(e_2, o_2) \\ &\wedge OBJ_LAB(e_3, o_3) \wedge STA_LAB(e_3, s_3) \\ &\wedge EQ_TO(o_1, o_3) \wedge EQ_TO(o_2, s_3) \\ &\rightarrow POSI_S(e_3, e_1) \wedge POSI_O(e_3, e_2) \end{aligned} \quad (18)$$

where $RObject(x)$ denotes the category information of x is position, $Position(x)$ denotes the category information of x is position, $POSI_S(x, y)$ denotes the subject of x is y and the category information of x is position, $POSI_O(x, y)$ denotes the object of x is y and the category information of x is position.

The rule for reasoning the relation between power source and research object is as follows

$$\begin{aligned} &Object(e_1) \wedge PSource(e_2) \wedge OBJ_LAB(e_1, o_1) \\ &\wedge OBJ_LAB(e_2, o_2) \wedge EQ_TO(o_1, o_2) \\ &\rightarrow HAS_PSOUR(e_1, e_2) \end{aligned} \quad (19)$$

where $PSource(x)$ denotes the category information of x is power source, $HAS_PSOUR(x, y)$ denotes x is the power source of y .

As the overall state entity is usually not given in the text, we use the default logic, if an object is a research object, then it is assumed that it has a overall motion. The rule can be written as

$$\begin{aligned} &Object(e_1) \wedge OBJ_LAB(e_1, o_1) \wedge STA_LAB(e_1, s_1) \\ &\rightarrow HAS_ORS(e_1, ORS_{o_1-s_1}) \end{aligned} \quad (20)$$

where $HAS_ORS(x, y)$ denotes x has a overall state y .

Relations between overall state and states details the movement process of research object, the rule can be expressed as

$$ORState(e_1) \wedge State(e_2) \wedge OBJ_LAB(e_1, o_1) \wedge OBJ_LAB(e_2, o_2) \rightarrow HAS_STATE(e_1, e_2) \quad (21)$$

where $ORState(x)$ denotes the category information is overall state, $State(x)$ denotes the category information of x is state, $HAS_STATE(x, y)$ denotes x has a state y .

The rule for reasoning relations between states and conditions is as

$$State(e_1) \wedge CON(e_2) \wedge OBJ_LAB(e_1, o_1) \wedge OBJ_LAB(e_2, o_2) \wedge STA_LAB(e_1, s_1) \wedge STA_LAB(e_2, s_2) \wedge EQ_TO(o_1, o_2) \wedge EQ_TO(s_1, s_2) \rightarrow HAS_CON(e_1, e_2) \quad (22)$$

where $CON(x)$ denotes the category information of x is condition (including displacement, acceleration, etc.), $HAS_CON(x, y)$ denotes state x has a condition (including HAS_DISP , HAS_ACC , etc.) y .

Finally, relations between objects and conditions details some properties of objects, including mass, efficiency, etc. The rule can be written as

$$OBJ(e_1) \wedge Condition(e_2) \wedge OBJ_LAB(e_1, o_1) \wedge OBJ_LAB(e_2, o_2) \wedge EQ_TO(o_1, o_2) \rightarrow HAS_CON(e_1, e_2) \quad (23)$$

where $OBJ(x)$ denotes the category information of x is object (including research object, reference object and power source).

2) HIDDEN RELATION MINING

Hidden information is indispensable for solving a mechanics problem. This article divides hidden information into two categories. The first category is general knowledge which aims to complete the comprehension model and the second is mechanics hidden knowledge for listing equations.

The general knowledge mainly store inherent properties or default conditions of objects. For example, without special declaration, a car (sports car, truck, etc.) usually move on road (ground, highway, etc.), an airplane or aircraft usually fly on the sky and is subject to air resistance, turning off the engine (break, decelerate, etc.) means that the moving object will maintain uniform deceleration linear motion and will eventually come to static, etc. This type of knowledge is related to objects or actions. As the objects and actions are usually fixed, we use knowledge graph to store these general knowledge. And general knowledge of P2 are provided in FIGURE 7.

The other type hidden information is mechanics hidden knowledge. For example, the final velocity of current state is equal to the initial velocity of the next state, the result force of ULM is $0N$, the acceleration of static is $0m/s^2$, etc. This type of knowledge is related to mechanics noun. As the number of

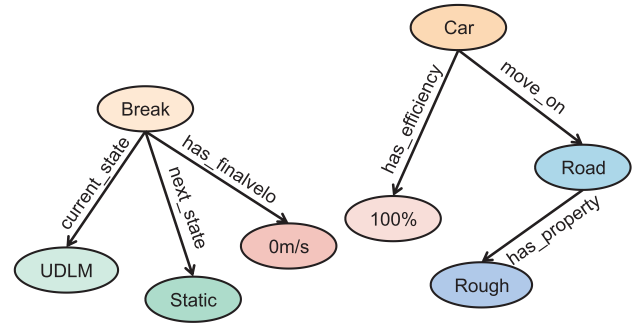


FIGURE 7. The general knowledge storage details of P2, where the ellipse represents the instance (subject or object), the directed line represent the predicate, the text next to the directed line indicates the predicate name, the ellipse connected to the arrow is the object, the ellipse connected to the nock is the subject, UDLM is abbreviation of uniform deceleration linear motion.

TABLE 1. The mechanics hidden knowledge mining algorithm, where $P(s, o)$ denotes s has a relation P with o .

Algorithm: Mechanics Hidden Knowledge Mining.
Input: The named entity list NE , label list LAB .
Output: The mechanics hidden knowledge list MHK .
1: Initialize $MHK = \emptyset$, $i = 0$, type list $T = \emptyset$, object list $O = \emptyset$, state list $S = \emptyset$.
2: Get l , where l denotes the length of NE .
3: for L in LAB do
4: Get $I_{t,e}$, $I_{o,e}$ and $I_{s,e}$, where $I_{t,e}$ is the type information of e , $I_{o,e}$ denotes the object information of e , $I_{s,e}$ denotes the state information of e .
5: $T \leftarrow I_{t,e} \cup T$, $O \leftarrow I_{o,e} \cup O$, $S \leftarrow I_{s,e} \cup S$.
6: end for
7: while $i < l$ do
8: if the i -th element of T is $STATE$ and the i -th element of NE is $Static$ then
9: $MHK \leftarrow HAS_RFORCE(I_{t,e}, I_{o,e}, I_{s,e}, 0N) \cup MHK$, $MHK \leftarrow HAS_IVELO(I_{t,e}, I_{o,e}, I_{s,e}, 0m/s) \cup MHK$, $MHK \leftarrow HAS_FVELO(I_{t,e}, I_{o,e}, I_{s,e}, 0m/s) \cup MHK$, $MHK \leftarrow HAS_DISP(I_{t,e}, I_{o,e}, I_{s,e}, 0m) \cup MHK$, $MHK \leftarrow HAS_ACC(I_{t,e}, I_{o,e}, I_{s,e}, 0m/s^2) \cup MHK$.
10: end if
11: if the i -th element of T is $STATE$ and the i -th element of NE is ULM then
12: $MHK \leftarrow HAS_RFORCE(I_{t,e}, I_{o,e}, I_{s,e}, 0N) \cup MHK$, $MHK \leftarrow HAS_ACC(I_{t,e}, I_{o,e}, I_{s,e}, 0m/s^2) \cup MHK$.
13: end if
14: $i = i + 1$
15: end while
16: Get s , where s denotes the number of state.
17: while $i < s - 1$ do
18: $MHK \leftarrow EQ_TO(FVelo_{I_{o,e}, I_{s,e}}, IVelo_{I_{o,e}, I_{s,e}} + 1) \cup MHK$.
19: $i = i + 1$
20: end while

state of an input problem is not fixed, we employ reasoning to mine these knowledge. The mining algorithm is provided in TABLE 1.

Nevertheless, there may be a conflict between direct information and mined hidden information. Thus, we borrow the idea of default logic to set different priorities for direct information and mined information by setting a conflict predicate set. We stipulated that the priority of direct information is higher than that of mined hidden information, and the conflict

TABLE 2. The given information of P2, where *HAS_ORIS* denotes has overall state, *HAS_STA* denotes has state, *HAS_ACC* denotes has acceleration, *HAS_MTI* denotes has motion time, *HAS_VELO* denotes has velocity, *HAS_TI* denotes has time, *HAS_DISP* denotes has displacement, *HAS_EFFI* denotes has efficiency, *HAS_PROP* denotes has property, *CUR_STA* denotes current state, *NEXT_STA* denotes next state, *HAS_GRA* denotes has gravity, *HAS_RFORCE* denotes has result force, *HAS_IVELO* denotes has initial velocity, *HAS_FVELO* denotes has final velocity, *EQ_TO* denotes equal to.

Direct Information	Hidden Information General Knowledge	Mechanics Hidden Knowledge
<i>HAS_ORIS</i> (<i>OBJ_1_0</i> , <i>ORS_1_OR</i>)	<i>IS_ON</i> (<i>OBJ_1_0</i> , <i>ROAD</i>)	<i>HAS_RFORCE</i> (<i>STATE_1_1</i> , 0 <i>N</i>)
<i>HAS_STA</i> (<i>ORS_1_OR</i> , <i>STATE_1_1</i>)	<i>HAS_EFFI</i> (<i>OBJ_1_0</i> , 100%)	<i>HAS_IVELO</i> (<i>STATE_1_1</i> , 0 <i>m/s</i>)
<i>HAS_STA</i> (<i>ORS_1_OR</i> , <i>STATE_1_2</i>)	<i>HAS_PROP</i> (<i>ROAD</i> , <i>ROUGH</i>)	<i>HAS_FVELO</i> (<i>STATE_1_1</i> , 0 <i>m/s</i>)
<i>HAS_STA</i> (<i>ORS_1_OR</i> , <i>STATE_1_3</i>)	<i>CUR_STA</i> (<i>ACT_1_4</i> , <i>UDLM</i>)	<i>HAS_ACC</i> (<i>STATE_1_1</i> , 0 <i>m/s²</i>)
<i>HAS_STA</i> (<i>ORS_1_OR</i> , <i>ACT_1_4</i>)	<i>NEXT_STA</i> (<i>ACT_1_4</i> , <i>Static</i>)	<i>HAS_DISP</i> (<i>STATE_1_1</i> , 0 <i>m</i>)
<i>HAS_ACC</i> (<i>STATE_1_2</i> , <i>ACC_1_2</i>)	<i>HAS_FVELO</i> (<i>ACT_1_4</i> , 0 <i>m/s</i>)	<i>HAS_RFORCE</i> (<i>STATE_1_3</i> , 0 <i>N</i>)
<i>HAS_MTI</i> (<i>STATE_1_2</i> , <i>MTI_1_2</i>)	<i>HAS_GRA</i> (<i>OBJ_1_0</i> , <i>GRA_1_1</i>)	<i>HAS_ACC</i> (<i>STATE_1_3</i> , 0 <i>m/s²</i>)
<i>HAS_MTI</i> (<i>STATE_1_3</i> , <i>MTI_1_3</i>)		<i>HAS_RFORCE</i> (<i>STATE_1_5</i> , 0 <i>N</i>)
<i>HAS_VELO</i> (<i>STATE_1_3</i> , <i>VELO_1_3</i>)		<i>HAS_IVELO</i> (<i>STATE_1_5</i> , 0 <i>m/s</i>)
<i>HAS_ACC</i> (<i>ACT_1_4</i> , <i>ACC_1_4</i>)		<i>HAS_FVELO</i> (<i>STATE_1_5</i> , 0 <i>m/s</i>)
<i>HAS_TI</i> (<i>ORS_1_OR</i> , <i>TIME_1_OR</i>)		<i>HAS_ACC</i> (<i>STATE_1_5</i> , 0 <i>m/s²</i>)
<i>HAS_DISP</i> (<i>ORS_1_OR</i> , <i>DISP_1_OR</i>)		<i>HAS_DISP</i> (<i>STATE_1_5</i> , 0 <i>m</i>)
		<i>EQ_TO</i> (<i>FVelo_1_1</i> , <i>IVelo_1_2</i>)
		<i>EQ_TO</i> (<i>FVelo_1_2</i> , <i>IVelo_1_3</i>)
		<i>EQ_TO</i> (<i>FVelo_1_3</i> , <i>IVelo_1_4</i>)
		<i>EQ_TO</i> (<i>FVelo_1_4</i> , <i>IVelo_1_5</i>)

resolution rule is

$$\begin{aligned}
 & DIRECT(t_1) \wedge HAS_SUB(t_1, s_1) \wedge HAS_PRE(t_1, p_1) \\
 & \wedge MINED(t_2) \wedge HAS_SUB(t_2, s_2) \\
 & \wedge HAS_PRE(t_2, P_2) \wedge SAME(p_1, p_2) \\
 & \wedge SAME(s_1, s_2) \rightarrow DEL_TRI(t_2) \quad (24)
 \end{aligned}$$

where *DIRECT*(*t*₁) denotes triple *t*₁ is belong to direct information, *HAS_SUB*(*t*₁, *s*₁) denotes the subject of *t*₁ is *s*₁, *HAS_PRE*(*t*₁, *p*₁) denotes the predicate of *t*₁ is *p*₁, *MINED*(*t*₂) denotes triple *t*₂ is belong to mined information, *HAS_SUB*(*t*₂, *s*₂) denotes the subject of *t*₂ is *s*₂, *HAS_PRE*(*t*₂, *p*₂) denotes the predicate of *t*₂ is *p*₂, *SAME*(*s*₁, *s*₂) denotes *s*₁ and *s*₂ are the same, *SAME*(*p*₁, *p*₂) denotes *p*₁ and *p*₂ are the same, *DEL_TRI*(*t*₂) denotes *t*₂ is in deleted information set. Finally, the triple in deleted information set will not used for the analysis and calculation.

This work acquire all the given conditions through five steps. First, the solver employs a predicate reasoner to extract the direct conditions. Second, a query is employed for acquiring the general knowledge conditions of input problem according to object entities and action entities. Third, updating given conditions by deconflicting the direct conditions with the general knowledge. Fourth, the mining algorithm is used for acquiring mechanics hidden conditions. Finally, the final given conditions are obtained by combining the mechanics hidden conditions with the given conditions in step 3.

According to the above procedure, the recognized entity and corresponding label of P2 are: **static** (*STATE_1_1*), **uniformly accelerated linear motion** (*STATE_1_2*), **0.5m/s²** (*ACC_1_2*), **uniform linear motion** (*STATE_1_3*), **20s** (*MTI_1_2*), **10s** (*MTI_1_3*), **brakes** (*ACC_1_4*), **-2m/s²** (*ACC_1_4*), **speed** (*VELO_1_3*), **36s** (*TIME_1_OR*), **displacement** (*DISP_1_OR*), where the bold word indicates the entity and the italic word in brackets indicates the label. TABLE 2 provides the total given information of P2.

TABLE 3. The procedure of force analysis.

Procedure: Force Analysis
Input: Given conditions, Rule Base.
Output: Predicates, Equations.
1 : for each state do
2 : Compute the number of forces on the research object, <i>m</i> ;
3 : initialize <i>i</i> = 0;
4 : while <i>i</i> < <i>m</i> do
5 : if <i>f_i</i> satisfies the decomposition conditions then
6 : Generate the component force predicate of <i>f_i</i> ;
7 : elif <i>f_i</i> does not satisfies the decomposition conditions then
8 : Generate the force predicate of <i>f_i</i> ;
9 : end if
10 : <i>i</i> = <i>i</i> + 1
11 : end while
12 : Generate the synthetic force predicate;
13 : Generate equations according to predicates.
14 : end for

C. PREDICATE LOGIC BASED SOLUTION REASONING

1) FORCE ANALYSIS

Solving dynamic problems is mainly to analyze the relations of forces on the object and to express these relationships with equations. The hidden information of dynamics problems is mainly the unannounced of friction and pressure. We exploit the position between objects and reasoning to mine these information. For example, if a car in on a road, then the car is subject to the friction of the road through reasoning. Additionally, if a box is move on a desktop and the dynamic friction factor between the box and the desktop is not equal to zero (default knowledge), then the box is subject to the friction of the desktop through reasoning. This work sets these position information as default knowledge and proposed an predicate logic-based method for force analysis. As provided in TABLE 3, the procedure is mainly divided into three steps: judging the number of forces, force decomposition and force synthesis.

For example, the description of problem 3 (P3) is “There is a wooden box on the level ground. The mass of the wooden box is *m* = 20kg, and the coefficient of kinetic friction

TABLE 4. The pool of calculation predicates and templates built in this paper.

Predicate	Template	Explicate
<i>EQ_TO(A, B)</i>	$A = B$	Variable A is equal to variable B .
<i>LEFT_ROTATE_QUATER(A, B)</i>	$A = 90 + B$	Angle A is equal to angle B plus 90.
<i>LEFT_ROTATE_HALF(A, B)</i>	$A = 180 + B$	Angle A is equal to angle B plus 180.
<i>RIGHT_ROTATE_QUATER(A, B)</i>	$A = 270 + B$	Angle A is equal to angle B plus 270.
<i>ADD_EQ(A, B, C)</i>	$A = B + C$	Variable A is equal to variable B plus variable C .
<i>MINUS_EQ(A, B, C)</i>	$A = B - C$	Variable A is equal to variable B minus variable C .
<i>MULTY_EQ(A, B, C)</i>	$A = B * C$	Variable A is equal to variable B multiply variable C .
<i>DIVID_EQ(A, B, C)</i>	$A = B / C$	Variable A is equal to variable B divided by variable C .
<i>SINE_EQ(A, B)</i>	$A = \sin(B)$	Variable A is equal to the sine function of B .
<i>COSINE_EQ(A, B)</i>	$A = \cos(B)$	Variable A is equal to the cosine function of B .
<i>TANGENT_EQ(A, B)</i>	$A = \tan(B)$	Variable A is equal to the tangenten function of B .
<i>LSEE_THAN(A, B)</i>	$A < B$	Variable A is less than variable B .
<i>MORE_THAN(A, B)</i>	$A > B$	Variable A is more than variable B .
<i>SUM(A, #)</i>	$A+ = \#$	Variable A is equal to the sum of all subsequent variables.

between the wooden box and the ground is $u = 0.20$. A child pushes the wooden box to the right with a thrust F to make a uniform linear motion, take $g = 10m/s^2$, what is the sliding friction on the wooden box.”

The recognized entities of P3 are: **wooden box** (*OBJ_1_0*), **on** (*POS_1_1*), **ground** (*ROBJ_1_1*), **20kg** (*MASS_1_0*), **0.2** (*FFAC_1_1*), **child** (*AOBJ_1_1*), F (*FORCE_1_1*), **right** (*MDIR_1_1*), **uniform linear motion** (*STATE_1_1*), **10m/s2** (*GACC*), **sliding friction** (*FRIC_1_1*), where the bold word indicates the entity and the italic word in brackets indicates the label.

Due to space reasons, the force analysis of P3 is given in Attached Table 1. In this article, we define calculation predicates and employ a template based method transform predicate to equations for computing the answer. TABLE 4 provides the calculation predicates and templates used in this work.

2) MOTION ANALYSIS

Different from existing studies, physical formulas are used as rules for solving problems in this article. The names, units, and symbols of the variables which make up the formula, the specific form and the application scenarios of the formula are essential for applying meta information based reasoning algorithm. We employ knowledge graph to store these physical rules. For example, the rule that sliding friction is equal to the product of pressure and friction factor can be stored as the schema provided in FIGURE 8.

From FIGURE 8 we can see that there are four predicates and six objects of rule 10. The content of the formula is $F_f = F_N * u$, which means that the friction is equal to the product of presure and the friction factor. Moreover, the number of rule 10 is P10 which denotes the 10-th physical rule. Additionally, the application scenario of rule 10 is FA which is the abbreviation of force analysis. Finally, there three variables: friction, presure and friction factor in the formula. The symbol of friction is F_f and the unit is N which indicates Newton, the symbol of presure is F_N and the unit is N too, the symbol of friction factor is u and the unit is null which indicates that u is just a coefficient.

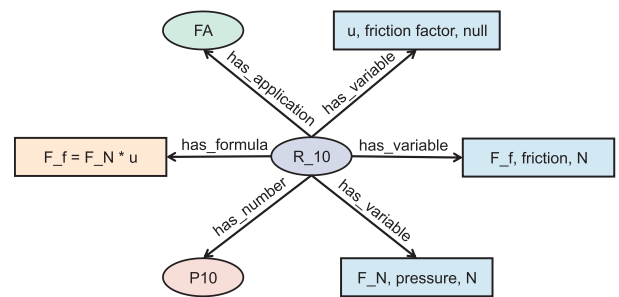


FIGURE 8. The storing schema of a physical rule, where **has_application** and **has_number** are object property, **has_variable** and **has_formula** are datatype property, ellipses denote instances, rectangles denote data instance expressed by s string, FA denotes force analysis, the object of **has_variable** consists of symbols, variable names, and units, separated by commas.

Current works employ the forward reasoning algorithm for computing solutions, which is space-consuming and produces useless conditions for complex mechanics problems. Zhang et al [21] proposed a meta information based backward reasoning algorithm to select equations for calculation. However, the algorithm needs to store the same formula with different objects information or states information separately, which consumes a lot of storage space. Therefore, we proposed a reasoning algorithm based on their work. The reasoning method can be divided into two steps, equation generating and equation selecting. The details are provided in TABLE 5.

To choose the suitable equations, we propose the concept of reasoning loss to constraint the reasoning direction. We stipulate that the smaller the reasoning loss of the current step, the more correct the reasoning direction. Define the reasoning cost as

$$\begin{cases} cost = I_i(Q_i, R) - S_i(C_i, R), \\ Q_i = IQ_{i-1}(Q, C, R) \\ C_i = IC_{i-1}(Q, C, R) \end{cases} \quad (25)$$

where Q denotes the question, C denotes the given condition, R denotes the rule $cost$ denotes the reasoning cost of question Q , condition C and rule R , $I_i(Q_i, R)$ denotes the i -th order initial reasoning loss of question Q and rule R , $S_i(C_i, R)$

TABLE 5. The meta information based equation selection algorithm.

Algorithm: Equation Generating and Selection.
Input: Tagged Sequence TS , Rule Base RB , Entity to Application Mapping $E2A$, Given Conditions GC , Hidden Equation Set HE , Force Equation Set FE , Question Q .
Output: Selected Equation SE .
1 : /* Equation Generating */
2 : Initialize $S = \emptyset, UR = \emptyset$
3 : for each E in TS do
4 : if the type information of E is $STATE$ then
5 : Get object information of E denote as o , state information of E denote as s .
6 : Generate variables of current state and marked by o and s .
7 : Get application of E , $A = E2A(E)$.
8 : for each R in RB do
9 : if application of R is A then
10 : Replacing variables in R with generated variables
11 : $UR \leftarrow R \cup UR$
12 : end if
13 : end for
14 : end if
15 : end for
16 : $UR \leftarrow HE \cup UR$
17 : $UR \leftarrow FE \cup UR$
18 : /* Equation selection */
19 : Initialize $RP = \emptyset$, max reasoning step ms , current step $s = 0$
20 : while $s < ms$ do
21 : Initialize $C = \emptyset$
22 : for each R in UR do
23 : if Q in meta information of R then
24 : Compute the reasoning cost of GC , Q and R , denote as c
25 : Update $C \leftarrow c \cup C$
26 : end if
27 : Get the minimum reasoning cost c_m
28 : if $c_m = 0$ then
29 : Update $SE \leftarrow R \cup SE, s = ms$
30 : end if
31 : if $c_m > 0$ then
32 : $SE \leftarrow R \cup SE, s+ = 1$
33 : Get new question according to C and R
34 : end if
35 : end for
36 : end while

denotes the i -th order reasoning score of given condition C and rule R , IQ_{i-1} denotes the $i - 1$ -th question reasoning function and IC_{i-1} denotes the $i - 1$ -th given condition reasoning function. Moreover, when there are multiple rules with the smallest reasoning cost, the algorithm will calculate a high-order loss of the current rule until there is only one rule has the smallest cost.

III. EXPERIMENTS

A. EXPERIMENTAL DATA

1) PROBLEM CLASSIFICATION

We divided the mechanics problems into three categories according to the test point: Newton’s laws, linear motion, power & energy. TABLE 6 provides examples of each category. Linear motion problems refer to problems only require motion analysis. These problems do not require force analysis and do not involve energy and power, Newton’s law problems refer to problems require both force and motion analysis but do not involve power and energy. Power and energy problems refer to the problems adding the knowledge of energy and power to the Newton’s law problem.

TABLE 6. Examples of each category, including Linear Motion, Newton’s Laws, Power & Energy.

Category	Example
Linear Motion	A car is driving at a constant speed on a straight highway with a speed of $v = 108\text{ km/h}$. Due to the danger ahead, the driver brakes urgently. When braking, the acceleration of the car is 5 m/s^2 , then the car closes the accelerator for 1.0 s . What is the distance to glide inside?
Newton’s Laws	There is a metal block with a mass of $m = 1\text{ kg}$ on the horizontal ground. The kinetic friction coefficient between it and the horizontal ground is $\mu = 0.20$. Under the action of the horizontal pulling force $F = 5\text{ N}$, it starts to move in a straight line with uniform acceleration to the right from rest, and g is 10 m/s^2 . find: (1) the acceleration of the metal block in the uniform acceleration motion; (2) the speed of the metal block starting to move 1.0 s from rest.
Power & Energy	Lift a 20 kg object vertically by 4 m at an acceleration of 2 m/s^2 from a standstill. find (1) the work W done by the pulling force on the object; (2) the average power $P1$ of the pulling force; (3) the amount of instantaneous power $P2$ of the pulling force when it reaches a height of 4 m . (g takes 10 m/s^2)

TABLE 7. The properties of training data, including training set, validation set and test set. Character Num denotes the total number of characters in the set, Entity Num denotes the total number of entities in the set, Problem Num denotes the total number of problems in the set, Ave characters denotes the average number of characters per problem in the set.

Project	Train Set	Validation Set	Test Set
Character Num	7838489	1019727	988392
Entity Num	462254	59931	60134
Problem Num	60033	7504	7505
Ave characters	130.57	135.89	131.70

2) TRAINING DATA

We collect training data through the Internet ^{1 2} for the experiment. There are a total of 6822 mechanics problems in three categories, including 2816 problems of linear motion, 2234 problems of Newton’s laws, and 1772 problems of power & energy. However, due to the large number of labels involved in the experiments, and the huge amount of data is the basis of deep learning algorithm, we perform data enhancement by replacing some words in the problem with synonyms or changing some entity values. For example, replace car with bus, 10 m/s with 20 m/s , etc. Additionally, we expand 10 more problems on the basis of each problem and the number of total training data is 75042. Finally, the training data is divided into training set, validation set and test set with a ratio of 8 : 1 : 1. The properties of the training data are provided in TABLE 7.

3) TEST DATA

The test data are collected from teaching materials, exercise books and test papers. A total of 1416 pure text problems are acquired for testing, including 712 linear motion problems, 356 newton’s laws problems and 348 power & energy problems. Additionally, we divide these problems into three categories according to the solving variable: solving kinematics, solving dynamics and solving power. Solving kinematics denotes that the question is kinematics variables such as displacement, velocity and acceleration etc., solving dynamics denotes that the question is dynamics variables such as friction, pressure and pulling force etc., solving power denotes that the question is power variables such as power, kinetic energy and potential energy etc. The property of our test data are provided in TABLE 8.

¹ <https://zujian.xkw.com/>

² <https://zujian.21cnjy.com/>

TABLE 8. The properties of test data, where kinematics denotes problem solving kinematics variables, Dynamics denotes problem solving dynamics variables, Power denotes problem solving power variables.

Project	Kinematics	Dynamics	Power
Character Num	123293	40934	24203
Entity Num	6803	2538	1415
Problem Num	931	306	179
Ave characters	132.43	133.77	135.21

B. EVALUATION METRICS

We use three levels of comprehension to evaluate the proposed method: complete comprehension, partial comprehension and incomprehension. To better define the three type comprehension, we define two measures as follows:

- 1) Identifying all direct conditions of the input problem, that is, correctly identify the type, boundary of each entity and correctly determine the relationship type between entities.
- 2) Identifying all hidden conditions of the input problem, which means mining all hidden relationships of the input problem.

Then the three levels are defined by using the two measures:

Complete comprehension: if and only if the solver fulfills the above two measures;

Partial comprehension: if and only if the solver can acquire but not all direct relations and hidden relations;

Incomprehension: if and only if fails to acquire any direct relations and hidden relations.

Take P2 as an example, if the algorithm acquire all the relations in TABLE 2, the comprehension level of the algorithm to P2 is complete comprehension. Moreover, if the algorithm fails to acquire any relation in TABLE 2, the comprehension level of the algorithm to P2 is incomprehension. Finally, the comprehension level of the algorithm to P2 is partial comprehension in other cases.

C. EXPERIMENT: EVALUATION OF THE PROPOSED ALGORITHM

1) IMPLEMENT DETAILS

We collect 46533 words from textbooks, workbooks, and test papers as our lexicon for experiment. Additionally, an analysis on the matching between the entities and the word vocabulary is given to demonstrate the conflict matching problem. Defining the entity conflict rate (ECR) as the ratio of non-identical overlapping entity of a dataset matches with the lexicon [21]. The ECR of our experiment is 27.31%.

Considering the size of the lexicons, as the using of BERT base for Chinese, the character embedding dimension and word embedding dimension of our experiment are set to 768. The maximum number of training epochs is set to 60. Moreover, the optimizer is Adam [35] and the decay rate of learning rate is set to $1e-2$. Additionally, the attention head of GAT is set to 3, the number of GAT layer is set to 2, the input dimension, hidden dimension and output dimension are set to 768, 384 and 192. Finally, the back-propagation

algorithm is used to solve the optimal value of all trainable parameters. To prevent overfitting, we use an early stop mechanism during the training process. When the F1 score does not increase for 10 consecutive batches, the training is terminated.

2) PERFORMANCE EVALUATION

In this experiment, we aim to evaluate the comprehension level and the solving accuracy of the proposed algorithm. First, we train the model on the collected training data. Then, we use the test data to evaluate the performance of the model which performs best on test set of the training data. The evaluation metrics values are provided in TABLE 9.

Several observations can be made from TABLE 9. First, the comprehension level of all test problems was either complete comprehension or partial comprehension, that is, there were no problems that were incomprehension. This is a good result. Second, for linear motion problems, among the 712 problems, the comprehension level of 49 problems is partial comprehension, and the complete comprehension rate reaches 93.12%. Third, the proportion of complete comprehension of Newton's laws problems reached 85.11%. Among them, 20 of the 159 problems of solving kinematic variables were partial comprehension, and the complete comprehension rate reached 87.42%; 33 of the 197 problems of solving kinematic variables were partial comprehension, and the complete comprehension rate reached 83.25%. Fourth, 81 of the 348 power & energy problems have a partial comprehension level and a complete comprehension rate of 76.72%. Among them, 17 of the 60 solving kinematic variables were partially understood, 28 of the 109 dynamic variables were partially understood, and 36 of the 179 power problems were partially understood. Moreover, the Reason Acc column shows that all problems fully understood were answered correctly. Finally, with the increase of test points, the accuracy of the proposed method has shown a downward trend. There are two main reasons for this phenomenon. One is that with the increase of knowledge points, the categories of entities will also increase; secondly, most of Newton's laws problems and power & energy problems are given in the form of a combination of pictures and texts, the pure text data is not a lot. We will focus on developing algorithms to address these two defects in feature work.

D. EXPERIMENT: COMPARISON WITH OTHER ALGORITHMS

To further evaluate the proposed algorithm, we conduct a comparative experiment between our algorithm and some existing algorithms. In specifically, we exam these algorithms on the 1416 test problems we collected. However, since some of these algorithms are DL based method which can not output an analysis and readable solutions, we take the solving accuracy as the evaluation metrics. Additionally, we define the accuracy improvement ratio to intuitively compare the solving accuracy between different algorithms.

TABLE 9. The statistical experimental results, where category denotes the category of problem, Question type denotes the type of various questions, Problem num denotes the number of problems, False num denotes the number of incorrectly solved problems, Complete comp denotes the ratio of complete comprehension, Partial comp denotes the ratio of partial comprehension, Incomp denotes the ratio of incomprehension, Reason Acc denotes the reasoning accuracy.

Category	Question Type	Problem Num	False Num	Complete Comp(%)	Partial Comp(%)	Incomp(%)	Reason Acc(%)
Linear Motion	kinematics	712	49	93.12	6.88	0.00	100.00
	dynamics	-	-	-	-	-	-
	power	-	-	-	-	-	-
	total	712	49	93.12	6.88	0.00	100.00
Newton's Laws	kinematics	159	20	87.42	12.58	0.00	100.00
	dynamics	197	33	83.25	16.75	0.00	100.00
	power	-	-	-	-	-	-
	total	356	53	85.11	14.89	0.00	100.00
	kinematics	60	17	71.67	28.33	0.00	100.00
Power & Energy	dynamics	109	28	74.31	25.69	0.00	100.00
	power	179	36	79.89	20.11	0.00	100.00
	total	348	81	76.72	23.28	0.00	100.00
Total	-	1416	183	87.08	12.92	0.00	100.00

1) BASELINE ALGORITHMS

In addition to the proposed method, we choose five other algorithms, DNS [17], Math-EN [18], GTS [19], GTL [20] and DLR [21] to construct the experiment. DNS is the first DL based algorithm for solving word problem. Math-EN employs a equation normalization algorithm to reduce the target sample space. GTS is a goal-driven method, which completes the goals by decomposing expressions. GTL is a work that uses GNN for encoding and tree based decoder to generate formulas. DLR is the first work that integrating DL and predicate logic for solving kinematics problems.

2) PARAMETERS SETTING

The details of the implement parameters are provided in TABLE 10. Since the bert-base model is employed, the embedding dimension of DLR and our method are set to 768. The embedding dimension of DNS, Math-EN, GTS and GTL are set to 128. In addition, the maximum training epoch of our algorithm is set to 60, while DLR to 40, and other baselines are set to 80. Furthermore, due to the using of GNN, the batch size of our algorithm and GTL are set to 2. The batch size of DLR are set to 8 and other baseline algorithms are set to 32. Moreover, all the experimental algorithms set the Shuffle to True. Additionally, the initial learning rate of DLR is set to different parameters according to the fitting ability of each layer: 5e-5 for the BERT layer, 1e-3 for the LSTM layer, and 1e-2 for the CRF layer. The initial learning rate of the other baseline algorithms are set to 1e-3. Besides, the learning rate decay rate, minimum learning and dropout ratio are set to 1e-2, 1e-5 and 0.5 for all the algorithms. Finally, we employed the Adam optimizer to train all models.

3) MAIN RESULTS

The main results of the comparative experiment are provided in FIGURE 9. It can be observed that the proposed algorithm has a better performance compared to other baselines. First, for all 1416 test questions, the proposed method has achieved an accuracy over 80%, while the best of other algorithms reached over 70%. Second, for the 712 Linear Motion problems, in addition to DNS, the accuracy of other

TABLE 10. The parameters setting of the comparative experiment, where Emb Dim denotes embedding dimension, Epoch denotes the maximum epoch, Batch denotes batch size, Ini LR denotes initial learning rate, LR Dec denotes the decay ratio of learning rate, Min LR denotes minimum learning rate, Drop denotes the ratio of dropout.

Model	Emb Dim	Epoch	Batch	Shuffle	Ini LR	LR Dec	Min LR	Drop
DNS	128	80	32	True	1e-3	1e-2	1e-5	0.5
Math-EN	128	80	32	True	1e-3	1e-2	1e-5	0.5
GTS	128	80	32	True	1e-3	1e-2	1e-5	0.5
GTL	128	80	2	True	1e-3	1e-2	1e-5	0.5
DLR	768	40	8	True	BERT: 5e-5	1e-2	BERT: 1e-5	0.5
					LSTM: 1e-3		LSTM: 1e-5	
					CRF: 1e-2		CRF: 1e-5	
Proposed	768	60	2	True	BERT: 5e-5	1e-2	BERT: 1e-5	0.5
					GAT: 1e-4		GAT: 1e-5	
					LSTM: 1e-3		LSTM: 1e-5	

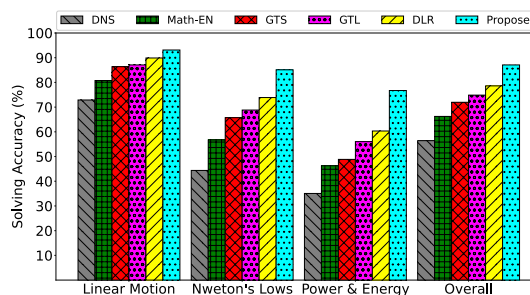


FIGURE 9. The solving accuracy of the proposed algorithm between baseline algorithms.

methods have exceeded 80%. The proposed algorithm has an accuracy of 90% which is the best among all the algorithms. Third, for the 356 Newton's Laws problems, the accuracy of each algorithm varies greatly, and the difference between the maximum accuracy and the minimum accuracy is close to 30%. Fourth, the performance difference of each algorithm is mainly reflected in the solving of the 348 Power & Energy problems. The best performing is nearly twice as accurate as the worst. The reason contributed to this result is the hidden information. Although DL based methods can learn to list formulas, the process is still a black box. Additionally, inherent general knowledge and hidden physical knowledge are difficult to learn through training. Solving a Newton's Laws problem or a Power & Energy problem need general knowledge and hidden physical knowledge, so the proposed algorithm significantly outperforms these methods in these two category.

TABLE 11. The accuracy improvement ratio between the proposed algorithm and baselines, where + denotes the proposed improves the baseline.

Category	DLR	GTL	GTS	Math-EN	DNS
Linear Motion	+3.59	+6.94	+7.80	+15.30	+27.75
Newton's Laws	+15.20	+23.67	+29.48	+50.00	+91.78
Power & Energy	+27.15	+36.93	+57.05	+65.85	+118.82
Overall	+10.79	+16.32	+21.01	+31.46	+54.32

TABLE 12. The ME, MV and SD of each algorithm.

	Proposed	DLR	GTL	GTS	Math-EN	DNS
ME	0.8594	0.7646	0.7161	0.6526	0.6107	0.5209
MV	0.8635	0.7730	0.7275	0.6465	0.6025	0.5215
SD	0.0303	0.0602	0.0634	0.0825	0.0873	0.0944

To compare the solving accuracy of the proposed method with baselines, we introduce a new evaluation metric. Define the accuracy improvement ratio as the ratio of the accuracy difference to the accuracy of the baseline algorithm, which can be written as

$$AIR = \frac{AO - AT}{AT} * 100\% \quad (26)$$

where *AIR* denotes the accuracy improvement ratio, *AO* denotes the accuracy of our method, *AT* denotes the accuracy of other algorithms. The results of accuracy improvement ratio between the proposed algorithm and baselines are provided in TABLE 11. DLR performs the best except for the proposed algorithm. Additionally, compared with DLR, the proposed method improves by 3.59% on linear motion problems, 15.20% on Newton's law problems, 27.15% on power and energy problems, and 10.79% overall.

E. SENSITIVITY ANALYSIS

It is well known that the DL methods are sensitive to variation in input data. To further evaluate the sensitivity, we perform a sensitivity experiment on the test data between the proposed algorithm and the baselines. We conducted a total of 10 sets of sensitivity experiments, and the test data of each set of experiments were 1000 questions randomly selected from 1416 test data.

The mean value (ME), median value (MV) and standard deviation (SD) of the solving accuracy of each algorithm are provided in Table 12. It can be observed that the solving accuracy of the proposed algorithm is noticeably better than other baselines. Additionally, the SD provided in the table can prove that the proposed method has better stability than other methods.

To statistically evaluate the performance of the proposed method, we also conduct a Wilcoxon signed rank test. We conducted a total of five groups of tests, each of which compared the proposed algorithm with one of the baselines. The null hypothesis of the test is that there is no significant difference between the two groups of data, whereas, the alternative hypothesis is that there is a significant difference between the two groups of data. The results of the Wilcoxon signed rank test are provided in Table 13. We can see that *p* values are significantly lower than 0.05 (5%). This is strong

TABLE 13. The results of the Wilcoxon signed rank test between the proposed algorithm and the baselines.

	DLR	GTL	GTS	Math-EN	DNS
<i>p</i> value (1-tail)	0.0068	0.0037	0.0027	0.0027	0.0027
<i>p</i> value (2-tail)	0.0135	0.0074	0.0054	0.0054	0.0054

evidence for rejecting the null hypothesis and accepting the alternative hypothesis. Combining the previous statistical results, we observe that the proposed method significantly outperforms baselines.

IV. DISCUSSION

We can observe that the proposed system achieve an accuracy of 87.08% (1233 of 1416) for solving mechanics problems. The proposed method simulates the human cognitive process, first understanding the input problem, and then finding the answer through reasoning. Additionally, as there is no complex syntactic analysis of the natural language, but the model information is integrated into the labels, the proposed method can also be applied to other languages. And only an ontology model and an general knowledge base are required to work on a different domain. We differentiate the reasons for the improvement of the proposed algorithm as follows.

- 1) The first is the architecture of the proposed method. Although has made critical breakthrough, current DL algorithm still in the status of perceptual intelligence. Whereas, people solve mechanics problems through systematic knowledge, and reasoning. However, perceptual intelligence cannot learn these knowledge efficiently. Therefore, obtaining various information in the problem first and then solving the problem through reasoning achieves better performance than solving the problem directly through a neural network.
- 2) General knowledge and hidden mechanics knowledge are also indispensable for solving a mechanics problem. Mechanics differs from circuits and mathematics in that it studies interactions and the motion of objects, which are related to real life. Therefore, comprehending a mechanics problem requires an understanding of the object properties. Moreover, the properties of some physical variables such as speed cannot be abruptly changed are particularly important in inferential calculations.
- 3) Lexical information is essential for Chinese sentences. Combining the pretrained model with graph neural network is an effective method to utilize the lexical information and contextual information of sentences.

V. CONCLUSION

In this work, we propose an intelligent tutorial algorithm for solving mechanics problems which fills the gap of intelligent tutoring in mechanics domain. Different from previous works, our method can solve both dynamics problems and kinematics problems combining neural network (NN), knowledge graph (KG) and reasoning. Moreover, a novel model combining pretrained model and graph attention

network is proposed for extract the information of input problem. Additionally, a hidden information mining algorithm is presented to supplement the conditions of input problem. Finally, a equation selection algorithm is introduced to reason the answer. The experimental results demonstrate the advantages of the proposed algorithm.

REFERENCES

- [1] W. L. and G. L. D. Zhang, "MathDQN: Solving arithmetic word problems via deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell., (AAAI)*, Feb. 2018, pp. 1–8.
- [2] D. Huang, S. Shi, C.-Y. Lin, and J. Yin, "Learning fine-grained expressions to solve math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 805–814.
- [3] S. Ughade and S. Kumbhar, "Survey on mathematical word problem solving using natural language processing," in *Proc. 1st Int. Conf. Innov. Inf. Commun. Technol. (IICIT)*, Apr. 2019, pp. 1–5.
- [4] J. Yang, "The research and design of humanoid solver for junior high school physics uniform motion calculation problems," Ph.D. dissertation, School Educ. Inf. Technol., East China Normal Univ., Shanghai, China, 2015.
- [5] X. Wenfei, "The design and development of a multiply representation based tutoring system for problem solving in high school physics," Ph.D. dissertation, Dept. Educ. Inf. Technol., East China Normal Univ., Shanghai, China, 2010.
- [6] W. Kintsch and J. G. Greeno, "Understanding and solving word arithmetic problems," *Psychol. Rev.*, vol. 92, no. 1, pp. 109–129, 1985.
- [7] D. G. Bobrow, "Natural language input for a computer problem solving system," Palo Alto Res. Center, CA, USA, Tech. Rep., 1964.
- [8] C. L. Chang and C. T. Lee, *Symbolic Logic and Mechanical Theorem Proving*. New York, NY, USA: Academic, 1973.
- [9] W.-K. Wong, S.-C. Hsu, S.-H. Wu, C.-W. Lee, and W.-L. Hsu, "LIM-G: Learner-initiating instruction model based on cognitive knowledge for geometry word problem comprehension," *Comput. Educ.*, vol. 48, no. 4, pp. 582–601, May 2007.
- [10] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, "Learning to solve arithmetic word problems with verb categorization," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 523–533.
- [11] S. Roy, T. Vieira, and D. Roth, "Reasoning about quantities in natural language," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 1–13, Dec. 2015.
- [12] B. He, X. Yu, P. Jian, and T. Zhang, "A relation based algorithm for solving direct current circuit problems," *Int. J. Speech Technol.*, vol. 50, no. 7, pp. 2293–2309, Jul. 2020.
- [13] P. Jian, C. Sun, X. Yu, B. He, and M. Xia, "An end-to-end algorithm for solving circuit problems," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 33, no. 7, Jun. 2019, Art. no. 1940004.
- [14] A. Mitra and C. Baral, "Learning to use formulas to solve simple arithmetic problems," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2016, pp. 2144–2153.
- [15] S. Roy and D. Roth, "Solving general arithmetic word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1743–1752.
- [16] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang, "Parsing algebraic word problems into equations," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 585–597, Dec. 2015.
- [17] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 845–854.
- [18] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to a expression tree," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1064–1069.
- [19] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 5299–5305.
- [20] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim, "Graph-to-tree learning for solving math word problems," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 3928–3937.
- [21] J. Zhang, J. Yuan, H. Guo, and X. Zan, "Integrating deep learning with first order logic for solving kinematic problems," *Int. J. Speech Technol.*, vol. 52, no. 10, pp. 11808–11826, Aug. 2022.
- [22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
- [23] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
- [24] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, Feb. 2022.
- [25] D. Kahneman, *Thinking, Fast Slow*. CA, USA: Farrar, Straus, 2011.
- [26] C. R. Perrault, *An Application of Default Logic to Speech Act Theory*. Cambridge, MA, USA: MIT Press, 1990.
- [27] Y. Zhang and J. Yang, "Chinese NER using lattice LSTM," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 1554–1564.
- [28] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.
- [29] T. Gui, Y. Zou, Q. Zhang, M. Peng, J. Fu, Z. Wei, and X. Huang, "A lexicon-based graph neural network for Chinese NER," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 1039–1049.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.
- [31] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," Aug. 2015, *arXiv:1508.01991*.
- [32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [33] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn. Berkshire, MA, USA, 2001*, pp. 1–9.
- [34] W. Wen-Tsun, "Basic principles of mechanical theorem proving in elementary geometries," *J. Automated Reasoning*, vol. 2, no. 3, pp. 221–252, Sep. 1986.
- [35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



JIARONG ZHANG received the B.S. degree in electrical engineering and automation and the M.S. degree in electrical engineering from Liaoning Technical University, Huludao, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree in electrical engineering from North China Electric Power University, China.

His research interests include intelligent information processing technology, deep learning, and intelligent education.



JINSHA YUAN received the M.E. degree in theoretical electrical engineering and the Ph.D. degree in electrical engineering and its automation from North China Electric Power University, Baoding, China, in 1987 and 1992, respectively.

He is currently a Professor and the Ph.D. Supervisor with North China Electric Power University. His research interests include intelligent information processing technology, wireless communication, and electromagnetic field numerical calculation method and application.



JIANING XU received the B.S. degree in electrical engineering and automation and the M.S. degree in electrical engineering from Liaoning Technical University, Huludao, China, in 2015 and 2018, respectively.

His research interests include intelligent information processing technology and deep learning.

• • •