Journal of *Marine Science and Engineering*

MDPI

Article

Study and Discussion on Computational Efficiency of Ice–Structure Interaction by Peridynamic

Yuan Zhang [1,2], Guoyang Zhang [1], Longbin Tao [3,4], Chao Wang [1,*], Liyu Ye [5], Shuai Sun [6] and Kang Han [1]

[1] College of Shipbuilding Engineering, Harbin Engineering University (HEU), Harbin 150001, China; zhangyuan@hrbeu.edu.cn (Y.Z.); guoyang.zhang@hrbeu.edu.cn (G.Z.); hankang3@163.com (K.H.)
[2] Department of Civil and Environmental Engineering, Faculty of Engineering, Norwegian University of Science and Technology (NTNU), 7034 Trondheim, Norway
[3] School of Naval Architecture and Ocean Engineering, Jiangsu University of Science and Technology, Zhenjiang 212003, China; longbin.tao@strath.ac.uk
[4] Department of Naval Architecture and Marine Engineering, University of Strathclyde, Glasgow G4 0LZ, UK
[5] Qingdao Innovation and Development Base of Harbin Engineering University, Harbin Engineering University, Qingdao 266400, China; yeliyuxrxy@hrbeu.edu.cn
[6] National Key Laboratory of Transient Physics, Nanjing University of Science and Technology, Nanjing 210094, China; shuai.s@njust.edu.cn
* Correspondence: wangchao0104@hrbeu.edu.cn

Abstract: The peridynamic (PD) theory is based on nonlocal mechanics and employs particle discretization in its computational domain, making it advantageous for simulating cracks. Consequently, PD has been applied to simulate ice damage and ice–structure interaction under various conditions. However, the calculation efficiency of PD, similar to other meshless methods, is constrained by the number of particles and the inherent limitations of the method itself. These constraints hinder its potential for further development in the field of ice−structure interaction. This study aims to explore the computational efficiency of various methods that can be employed to improve the computational cost of PD in ice–structure interactions. Specifically, we analyze the computational efficiency of three different methods (the MPI parallelization, the updated link−list search method, and the particle−pair method) and their collaborative calculation efficiency to reduce simulation time. These methods are employed to calculate ice–ship interaction, and their coupled efficiency is studied. Furthermore, this study discusses the computation strategy to improve efficiency on using the PD method to calculate ice–structure interaction. The present work provides scholars who employ PD to calculate ice–structure interaction or ice damage with a referential discussion plan to achieve an efficient numerical computation process.

Keywords: ice–structure interaction; computational efficiency; particle−pair method; MPI parallelization; peridynamic (PD)

Citation: Zhang, Y.; Zhang, G.; Tao, L.; Wang, C.; Ye, L.; Sun, S.; Han, K. Study and Discussion on Computational Efficiency of Ice–Structure Interaction by Peridynamic. *J. Mar. Sci. Eng.* **2023**, *11*, 1154. https://doi.org/10.3390/jmse11061154

Academic Editor: Sasan Tavakoli

Received: 25 April 2023
Revised: 27 May 2023
Accepted: 28 May 2023
Published: 31 May 2023

1. Introduction

The peridynamic (PD) theory is a nonlocal theory proposed by Silling in 2000 [1]. The theory reconstructs the motion equation of solid mechanics, and it can solve the differential equation in the form of integration. It has great advantages in simulating the objects with discontinuity without other assumptions and extra damage criteria, such as generation and propagation of cracks and a failure model of materials [2,3]. The PD theory has two primary forms: bone−based PD and state−based PD. In the bone−based PD theory, the force between two particles is set to the same in magnitude and opposite in direction, limiting Poisson's ratio of materials [1]. In the state−based PD theory, the magnitude and direction of the force between two particles can be different. It eliminates the limitation in Poisson's ratio of materials, making the PD theory applicable to elastic–plastic and incompressible materials [4].

At present, the research of peridynamic applications is developing rapidly. The PD method has been successfully applied to solid mechanics, and it's applications has been verified in the field of thermodynamics, electricity, and fluid mechanics [5–7]. PD also serves as a preferable tool for simulating various ice mechanics and ice interacting with an idealized structure. Wang et al. [8] studied ice fragmentation under explosive loading, which shows PD's ability in simulating ice elastic fracture under high pressure. This is followed by Zhang et al. [9], who studied ice thermoelastic fragmentation with fully coupled thermal equation in state−based PD. The in−plane and out−of−plane failures of a square ice sheet were investigated, and the results were compared with a splitting test by Vazic et al. [10]. Another ice simulation with PD also involves polycrystalline S2 ice [11], ice−sloping/vertical structure interaction [12,13], and deicing [14]. Though a lot of works on ice mechanics were studied, these studies pay more attention to the ice mechanical model of idealized boundary conditions and the failure mechanism of an ice constitutive model rather than the practical application in the engineering field. More efforts are still needed to realize the application of PD in the field of engineering forecasting.

In view of more engineering applications, the first study employing PD in ice crushing was conducted by Liu et al. [15], in which the ice plate was crushed by a rigid cylinder in the bond−based PD numerical framework. Following this work, the ship navigating in rubble ice was simulated by bond−based PD coupling with a Voronoi algorithm for the formation of pack ice [16]. Moreover, PD was also employed to simulate ice–ship interaction and ice–propeller interaction by introducing a new contact detection model between a mesh element and particles [17–19] and the PD−FEM coupling method [20]. The above work applied the bond−based PD method to the real operating conditions of marine structures with a fixed Poisson's ratio of 1/3. However, the discretization of ice particles is relatively rough, and the working conditions of the structure were assumed in a simple operating mode, which is mainly because the fast computational efficiency can be achieved by bond−based PD. Afterwards, in order to be able to simulate the fine structure model and the mechanism of ice action, as well as the ice constitutive model of the plastic behavior, a state−based model is urgently needed for the study of both 2D cases and 3D cases in complex and practical scenarios. Unfortunately, due to the limitation of low calculation efficiency, the engineering investigation of ice–structure interaction using the state−based PD model is an underdeveloped field. In addition, though the PD theory offers a better approach to deal with cracks and failure, the computational efficiency is prohibitive to obtain an excellent exact solution to engineering problems.

To improve the efficiency of PD simulations, researchers have conducted extensive algorithmic research. Diyaroglu et al. [21] introduced an effective method for searching the neighborhood particles of each material point using 2D and 3D local squares and cubes to allocate the region of each particle in advance. Vazic et al. [22] conducted a comprehensive study on various neighborhood particle search algorithms suitable for dynamic simulations, including brute−force search, region partitioning, balanced K−D tree algorithm, and updated R−tree algorithm based on packaging algorithm. They found that the R−tree data structure algorithm outperforms the other three algorithms, while brute−force is the most time−consuming method. Dominguze et al. [23] proposed a dynamic updating neighborhood search method based on Verlet list technology. By rearranging the particles in a manner that places adjacent cells' particles in close memory proximity, the methods' performance can be enhanced. PD can be easily parallelized by OpenMP, but this shared parallel mode puts higher demands on the hardware. A open−source software, Peridigm, enables MPI parallelization for the PD method, which greatly improves the possibility of the engineering application calculation of PD. Guo et al. [24] investigated a dynamic ice−milling process and structural response of a propeller blade profile by Peridigm. However, the update speed of Peridigm is far behind the progress of the current PD method. Zhang et al. [25,26] investigated a continuous icebreaking process by proposed MPI parallel strategies and compared icebreaking differences between two typical icebreaker bows. It can be seen that the calculation efficiency of the PD method is increasing with the progress

of research, but there is still a long way to go to reach the analysis of polar operations of structures under more complex conditions.

In response to the development deficiencies discussed above, this study aims to address the numerical efficiency of ice–structure interaction using PD, including the numerical methods that improve the computational time and their MPI parallel calculation with multiple threads. The present study contributes to provide a feasibility plan for the PD method applied in engineering fields. Note that the proposed strategies and discussions are not limited to ice–structure interaction (though the paper focuses on ice–structure interaction only), but it is easily possible to implement a high−efficiency strategy to other engineering fields following the provided material in this study.

The present study demonstrates the numerical efficiency of three methods to improve PD computing efficiency including the algorithm optimization method and parallelization. The paper is organized as follows: The challenges on PD theory simulating ice–structure interaction are briefly reviewed in Section 2. In Section 3, the numerical strategies of the updated link−list algorithm to accelerate the ice particle search, particle−pair method to accelerate the numerical integration, and the MPI parallel computing technology are presented. Their time efficiency is investigated compared with the conventional framework of the PD method. Moreover, collaborative speedup and the efficiency of the MPI parallelization among the particle−pair method, the updated link−list method, and the original PD algorithm are investigated in Section 4. Finally, an engineering event of icebreaker breaking level ice is studied by the proposed efficient numerical method, which better demonstrates the engineering application of the PD method in a parallelization scheme with updated methods. Section 6 discusses the relevant conclusions of this paper and efficient strategies for calculating ice–structure interaction.

2. Peridynamic Theory for Ice–Structure Interaction

Since the bond−based PD has been commonly introduced in simulating ice–structure interaction, this section provides a brief overview of the ordinary state−based PD, which serves as the numerical framework for the present study. Additionally, the challenge in simulating ice–structure interaction caused by low computational efficiency is discussed in Section 2.3, including large−scale simulations and fine structure models.

2.1. Ordinary State−Based Peridynamic for Ice Model

Peridynamic (PD) is a nonlocal theory that provides an alternative formulation for continuum mechanics. This method can solve differential equations in the form of integration, so it is suitable for discontinuous problems. The continuum is made up of an infinite number of particles at any one time. As shown in Figure 1, an ice body is discretized into a particle domain. Ice particles interact with each other in a certain range, δ, which is called horizon. All ice particles within the distance δ constitute the family members of ice material point i, and the volume range occupied by family members is defined as interact domain, H_x. In the PD theory, the displacement change between two particles in the deformed body can be expressed as follows [27]:

$$\boldsymbol{\eta} = \mathbf{u}' - \mathbf{u} = (\mathbf{y}_{(i)} - \mathbf{y}_{(j)}) - (\mathbf{x}_{(i)} - \mathbf{x}_{(j)}) \tag{1}$$

The vector $\mathbf{x}_{(i)}$ (or $\mathbf{x}$) represents the location of point i. The force density and the displacement are defined as $\mathbf{t}$, $\mathbf{u}$. Similarly, the relative physical quantities of material point j are $\mathbf{x}_{(j)}$ (or $\mathbf{x}'$), $\mathbf{t}'$, $\mathbf{u}'$. With respect to a Cartesian coordinate system, material point $\mathbf{x}_{(i)}$ experiences displacement, $\mathbf{u}$, and the new location of ice particle i is defined as $\mathbf{y}_{(i)}$. The relative position is expressed as $\mathbf{u} = \mathbf{x}_{(i)} - \mathbf{x}_{(j)}$, and the relative position after deformation is expressed as $\mathbf{u}' = \mathbf{y}_{(i)} - \mathbf{y}_{(j)}$.

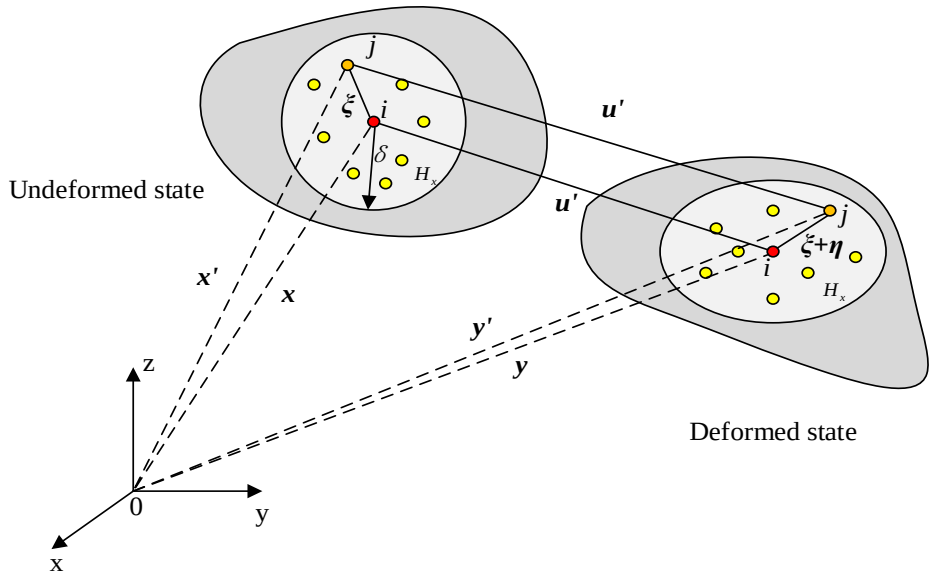


Figure 1. Ice discretization and ice particle i interact with those in sphere H_x (family members in horizon) through bonds.

In an ordinary state−based PD (OSB−PD), the governing equation is defined as follows [27]:

$$\rho_{(i)}\ddot{\mathbf{u}}(\mathbf{x},t) = \int_H \left(\mathbf{t}(\mathbf{u}'-\mathbf{u},\mathbf{x}'-\mathbf{x},t) - \mathbf{t}'(\mathbf{u}-\mathbf{u}',\mathbf{x}-\mathbf{x}',t)\right)\mathrm{d}H_x + \mathbf{b}(\mathbf{x},t) \tag{2}$$

where ρ represents the density of the ice, $\mathbf{b}(\mathbf{x},t)$ represents the external force of the ice body; $\mathbf{t}$ and $\mathbf{t}'$ represent the force density vectors of ice particles i and j, and $\ddot{\mathbf{u}}(\mathbf{x},t)$ represents the acceleration of material point i. The equation of force density is expressed as follows:

$$\begin{cases} \mathbf{t}(\mathbf{u}'-\mathbf{u},\mathbf{x}'-\mathbf{x},t) = \frac{1}{2}B_1\frac{\mathbf{y}'-\mathbf{y}}{|\mathbf{y}'-\mathbf{y}|} \\ \mathbf{t}'(\mathbf{u}-\mathbf{u}',\mathbf{x}-\mathbf{x}',t) = -\frac{1}{2}B_2\frac{\mathbf{y}'-\mathbf{y}}{|\mathbf{y}'-\mathbf{y}|} \end{cases} \tag{3}$$

where the parameters B_1 and B_2 are additional parameters for the state−based PD.

The ice model presented here is an isotropic, linear elastic material model, also commonly known as the linear peridynamic solid (LPS) material model. This model can also be extended to other types of material behavior, such as plasticity or viscoelasticity. The present study employs elastic ice behavior to demonstrate numerical results.

2.2. Contact Model for Ice–Structure Interaction

When dealing with ice–structure interaction, two main algorithms are employed to model contact force.

The first is contact force repelling short−range force, originally proposed by Silling [1,28] and applied to ship–ice contact by Liu et al. [16]. In this model, both ice and structure are discretized into particles, the force between ice particle and structure particle is:

$$\mathbf{f}_{contact} = -\frac{\mathbf{p}}{|\mathbf{p}|}c_{sh}\left(\frac{|\mathbf{p}|}{r_{sh}} - 1\right) \tag{4}$$

where $\mathbf{p}$ is the distance between ice particles and structure particles after deformation, c_{sh} is the short−range force constant, and r_{sh} is the critical distance, beyond which the short−range force does not exist.

The other contact model has a wider range of applications, can simulate any shape of the structure, and is also suitable for coupling with FEM [29]. In this model, the structure is

discretized into mesh elements. The velocity of the ice particle at time $t + \Delta t$ is obtained by the following formula:

$$\overline{\boldsymbol{v}}_{(i)}^{t+\Delta t} = \frac{\overline{\boldsymbol{u}}_{(i)}^{t+\Delta t} - \boldsymbol{u}_{(i)}^{t}}{\Delta t} \tag{5}$$

$\overline{\boldsymbol{u}}_{(i)}^{t+\Delta t}$ is the displacement of the newly allocated ice particle at time $t + \Delta t$, and $\boldsymbol{u}_{(i)}^{t}$ is the displacement of the ice particle at time t. At time $t + \Delta t$, the contact force of the ice particle on the structure can be calculated by the following formula [27]:

$$\boldsymbol{F}_{(i)}^{t+\Delta t} = -\rho_{(i)} \frac{\overline{\boldsymbol{v}}_{(i)}^{t+\Delta t} - \boldsymbol{v}_{(i)}^{t}}{\Delta t} V_{(i)} \tag{6}$$

where $\boldsymbol{v}_{(i)}^{t+\Delta t}$ is the velocity of the ice particle infiltrated into the structure body at time $t + \Delta t$, and $\rho_{(i)}$ and $V_{(i)}$ are the density and volume of the ice particle, respectively. At time $t + \Delta t$, the contact force on any structure surface element can be obtained by superimposing the contact force of all ice particles in contact with it:

$$\boldsymbol{F}_{j,contact}^{t+\Delta t} = \sum_{i=1} \boldsymbol{F}_{(i)}^{t+\Delta t} \lambda_{(i)}^{t+\Delta t} \tag{7}$$

where $\lambda_{(i)}^{t+\Delta t}$ can be defined by the following formula:

$$\lambda_{(i)}^{t+\Delta t} = \begin{cases} 1 & \text{contact with } j \text{ element of structure} \\ 0 & \text{no contact} \end{cases} \tag{8}$$

The second approach is adopted in this paper to demonstrate numerical efficiency. However, the discussion of computational efficiency in this paper applies to both methods.

2.3. Challenges in Engineering Application in View of Computational Cost

PD simulation plays a crucial role in studying ice's large deformation, failure mechanisms in various boundary conditions and ice–structure interaction modes; however, the computational cost associated with PD simulations can be prohibitive. Ice–structure interaction is a complicated process, and its engineering application challenges are mainly limited by the following aspects due to the low efficiency of PD.

- Difficult to model large−scale cases.

The calculation efficiency of PD depends directly on the number of particles; obviously, the larger the number of particles, the longer the calculation time. This computational cost is related to the neighborhood particle search time and the time integration step. Although the number of particles can decrease with an increasing scale when computing large−scale ice damage and ice–structure interaction problems, such as ice–ship collisions on the prototype scale and ice rheology in the geophysics, the calculation time will not be significantly improved with the reduction of the number of particles.

- Difficult to model fine and accurate shape structure breaking ice.

Taking ice–ship collision as an example, the outline characteristics of the bow greatly determine the icebreaking mode and icebreaking load. In order to explain the influence of the change of bow shape parameters on the icebreaking process, the accurate and fine bow model must be established. This makes a finer discretization of the ice domain and the hull, ensuring the capture of change of the bow parameters acting on the icebreaking. Subsequently, the number of particles will be greatly increased, and the time step size will be greatly reduced, which is a huge challenge for computational cost.

- Difficult to model long duration cases.

An explicit solver provides a guaranteed way to model ice damage, but the PD method requires a very small time step compared with the FEM in an explicit solver. A value for the maximum stable time step is determined as follows [30]:

$$\Delta t_{crit} = \sqrt{\frac{2\rho}{\sum_p \Delta V_p C_p}} \tag{9}$$

where ρ is the density, p iterates over all the neighbors of the given ice particle, ΔV_p is the volume associated with neighbor p, and C_p is the micromodules between the given ice particle and neighbor p. Δt_{crit} depends on the number of particles in the neighborhood; in other words, it depends on the horizon size and the dimension of the calculation case (for example, if the horizon size is 3.015, the summation times of horizon particles are 6, 25, and 122, corresponding to 1D, 2D, and 3D, respectively). FEM employs Courant–Friedrichs–Lewy to determine the time step. With same input model and setup, the time steps needed by PD are much smaller than those required by FEM. Therefore, when predicting the damage problem of ice over a long period of time, such as the time period reaching days or weeks, the computational cost increases sharply with the increase in the calculation period. If the time step cannot be increased appreciably, computational efficiency must be improved to achieve long−period ice–structure interaction prediction.

- Difficult to model ice–structure interaction in a complex working condition.

To the authors' knowledge, the PD has not been developed to model ice–structure interaction in a complex working condition, such as the ship motion in six degrees in breaking ice.

Through the understanding of the PD algorithm and the experience of predicting ice–structure interaction, we present three ways to improve the computational efficiency of ice–structure interaction. Additionally, the computational efficiency of each method is analyzed accordingly. They are:

- Updated link−list search method to accelerate particle search in the ice domain;
- Particle−pair method to accelerate time integration in solving an ice constitutive model;
- MPI parallelism to improve the efficiency of the entire algorithm.

These studies are presented in Section 3. Accordingly, collaborative computing with three methods and its efficiency are studied in Section 4.

It is worth noting that the research in this paper cannot completely solve the above−mentioned challenges. However, it is very meaningful to raise the above−mentioned challenges to provide thinking for the development of the industry and to call on more scholars to pay attention to the future development. Additionally, our work serves as an inspiring starting point for addressing the aforementioned problems, and there remains an ample scope for future research to build upon the findings of this study.

3. Computation Efficiency of Optimization Methods

3.1. Updated Link−List Search Method to Improve Family Member Search of Ice Domain

In the PD solution, the search for particles within the horizon plays a crucial role and is a time−consuming process, along with time integration. The time consumption is primarily dependent on the horizon distance δ. It is important to note that constructing the stiffness matrix for solving static problems in PD involves considering the family members of each material point. This approach leads to a higher density stiffness matrix compared with the finite element method (FEM) used in traditional mechanics. As a result, the computational time increases, leading to lower calculation efficiency. To address this issue and improve computational efficiency, this section introduces an updated and extended version of the link−list search method [31] from the smoothed−particle hydrodynamics (SPH) theory to PD solutions.

A schematic diagram of an updated link−list method is shown in Figure 2. A series of 3D square grids are arranged in the discrete ice domain, including the boundary particles. The side length of a square is set to be the same as the horizon length, δ. The search for horizon particles is only carried out in the current grid and adjacent grids. For example, as shown in Figure 2, the family particles of the particle in grid 1 only exist in grids 1, 2, 8, and 9; the family particles of the ice particle in grid 28 only exist in grids 20, 21, 27, 28, 34, and 35; by analogy, the family particle members of particles in grid 31 only exist in grids 23, 24, 25, 30, 31, 32, 37, 38, and 39. This way avoids the global search one by one in the entire PD. In the 1D, 2D, and 3D cases, the numbers of grids to be searched are 3, 9, and 27, respectively, indicating a great deal of savings in search computing time. For the specific numerical implementation process of the method, please refer to our previous work [32]. In general, it is necessary for the grid size to be greater than or equal to the horizon size in order to effectively search for all the family particles within the current grid and its adjacent grids. As the grid size increases, the number of search operations per step also increases. Therefore, the optimal grid size is the minimum value that guarantees the inclusion of all family particles in the horizon, which corresponds to the horizon size, δ.

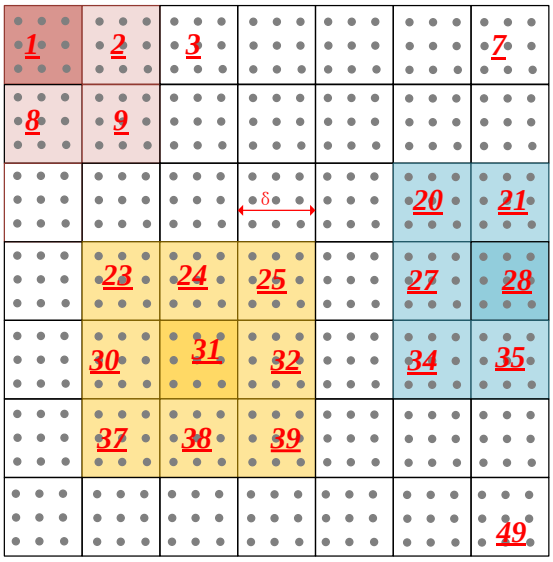


Figure 2. Schematic diagram of an updated link−list search method to search the family particle in the ice domain.

Here, we establish an ice domain with different discretization. The information for ice discretization is shown in Table 1. Then the family member search is carried out to check the efficiency. A comparison of time consumption in searching for family particles between traditional brute−force search and updated link−list search with different numbers of ice particles is given in Table 2. The time consumption of three computers with different configurations is also listed in the table as a reference.

Table 1. Information of ice domain discretization.

Conditions	Number of Particles	Number of Grids in Link−List Method
A	7000	304
B	64,000	1977
C	125,000	3564
D	216,000	6048
E	343,000	9396
F	512,000	14,040
G	729,000	19,872
H	1,000,000	19,683

Table 2. Time consumption of updated link−list search method (unit: s).

Conditions	Configuration 1		Configuration 2	
Type	Updated Link−List Method	Brute−Forced Method	Updated Link−List Method	Brute−Forced Method
A	0.01562	0.046785	0.03125	0.0625
B	0.23437	3.25	0.296875	4.1875
C	0.45312	12.6875	0.57812	18.96875
D	0.73437	35.25	1.03125	74.31812
E	1.48437	98.75	1.82812	221.96875
F	2.875	356.59375	2.75	460.14062
G	3.29687	724.140625	4.39062	957.45312
H	4.67187	1672.14062	6.4375	2667.89062

Configuration 1: Intel(R) Core (TM) i7−10875H @ 2.3 GHz, 32 GB RAM, MS Windows 11 × 64. Configuration 2: Intel(R) Core (TM) i5−8250U @ 1.6 GHz, 8 GB RAM, MS Windows 10 × 64.

Figure 3 illustrates a 100% stacked bar chart comparing the computational efficiencies of the brute−force search method and the updated link−list search method. The plot in Figure 4 also showcases the time consumption of both methods. It is evident that the updated link−list search method significantly reduces the computational cost of searching for family particles. Additionally, the updated link−list method exhibits superior computational efficiency as the number of particles increases, in comparison with the brute−force search method. In other words, as the number of particles increases, the rate at which the brute−force search method consumes time far exceeds that of the updated link−list method. For instance, at 7000 particles, the respective time consumptions for the two methods are 0.125 and 0.51562 s. However, at 1,000,000 particles, these values escalate to 25.29687 and 9077.875 s, respectively. Consequently, the growth multiples of time consumption are 202.37 for the updated link−list method and 17,605.74 for the brute−force search method.

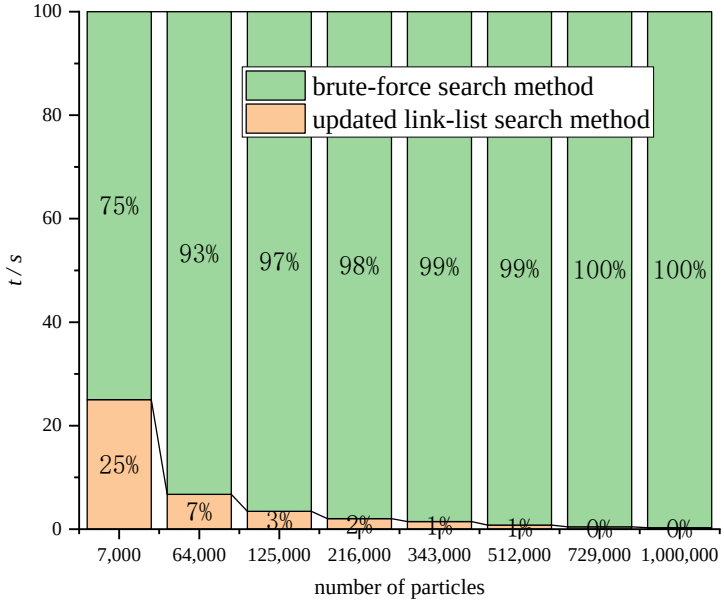


Figure 3. Comparison of updated link−list search and traditional brute−forced search under different configurations.

In Figure 4, it can be observed that the time cost of the updated link−list method exhibits a more consistent linear relationship with the increase in the number of particles in release mode. Furthermore, the growth trend is less influenced by changes in configuration when the number of particles is small. However, the impact of computer configuration on calculation time becomes evident as the number of particles increases significantly. This is

due to the fact that with a large number of particles, the computer's ability to store and process data can be clearly distinguished.

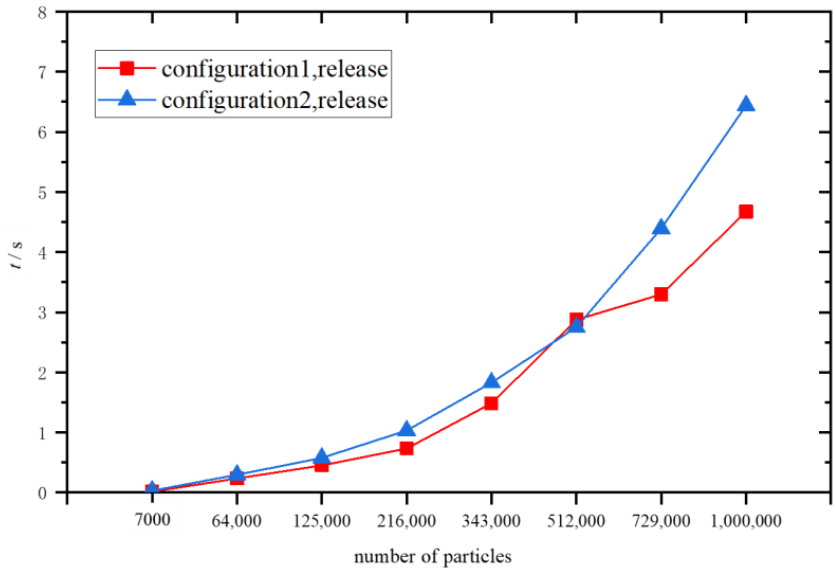


Figure 4. Time cost of updated link−list search in different configurations.

During the calculation process, the family particles searched need to be stored in a line list called *nodefam*; when the number of particles reaches 1,000,000, a stack overflow issue arises. This is because each thread has its own stack allocated for storing local variables and functions, depending on the compiler. To address this issue, allocatable arrays are defined, providing a temporary solution to the stack overflow problem. From this perspective, relying solely on algorithm improvements for serial computation cannot fully maximize the performance of the computer. Therefore, it remains crucial to implement a multithreaded parallel computation of the PD algorithm.

3.2. Particle−Pair Method to Accelerate Time Integration in Solving Ice Constitutive Model

In the original PD framework, the integration process is primarily performed on individual particles, which results in repeated search and calculation for each pair of interacting particles. Figure 5 provides an example of a 1D ice bar with 7 particles. In this figure, all the interacting ice particles are listed with a horizon value of 3. However, it can be observed that certain interacting particles are calculated redundantly, e.g., 1–2 and 2–1, 1–3 and 3–1, and 1–4 and 4–1. Specifically, in Figure 5, ice particles from 1 to 7 are stored in the array S, occupying 7 memory addresses. The interacting particles for each particle are stored in the array T in a specific order, occupying 30 memory addresses. Consequently, the original PD algorithm requires 30 loops for the integration calculation at each time step. However, it is evident that 15 of these loops are repetitive, as highlighted in Figure 5 with a diagonal background. To eliminate the redundancy in particle integration loops, a particle−pair algorithm is proposed. This algorithm transforms the integration process from individual particles to interacting particle pairs. For instance, the arrays S and T in Figure 5 can be replaced by a particle−pair array with two columns, represented as *pair (count, 1)* and *pair (count, 2)* (where count represents the total number of particle pairs with active relationships), as depicted in Figure 6.

It can be found that the particle−pair method mentioned above not only reduces the amount of family particle search (as shown in Figure 5: the computational amount of search and integration processes is reduced from 30 to 15), but also eliminates the need for defining large arrays. Only one array is required to replace three large arrays, S, T, and their corresponding pointer array. The particle−pair method retains the mathematical model of the PD theory, and the parameter settings for ice and the numerical setup remain the same

as in the original PD algorithm. The numerical implementation process of this method involves simply modifying the loop variables of each particle and its family particles using particle−pair array, which can be found in our previous work [33].

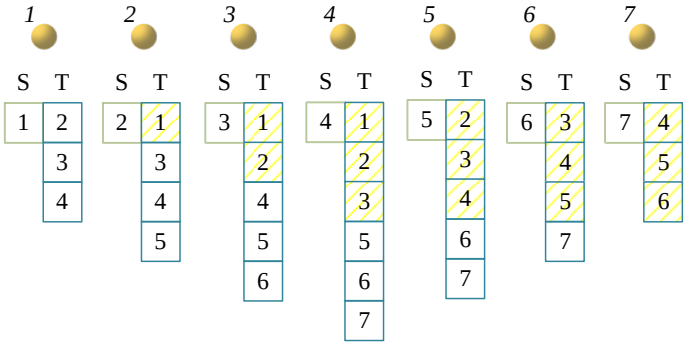


Figure 5. List of interacting particles in one dimension.

Adress:		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data:	*Pair(count,1)*	1	1	1	2	2	2	3	3	3	4	4	4	5	5	6
	Pair(count,2)	2	3	4	3	4	5	4	5	6	5	6	7	6	7	7

Figure 6. The array of particle pair refers to the case in Figure 5.

To validate the accuracy of the particle−pair method, an impact between a rigid cylinder and an ice plate is calculated. In this case, the ice is not allowed to be damaged. The calculation model and results, specifically the displacement of the ice plate in the y−direction, are presented in Figures 7 and 8. The accuracy of the results is verified by comparing them with those obtained from the original PD algorithm.

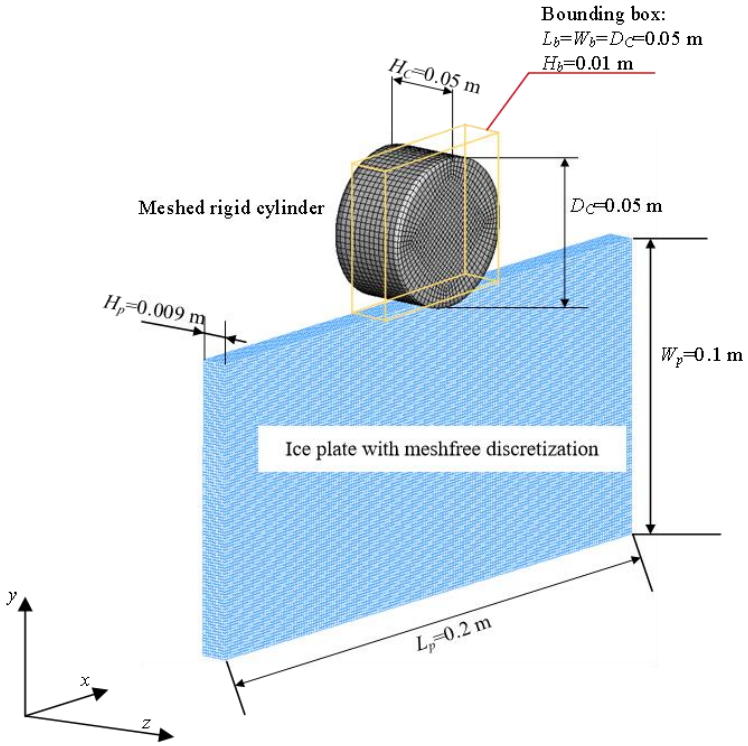
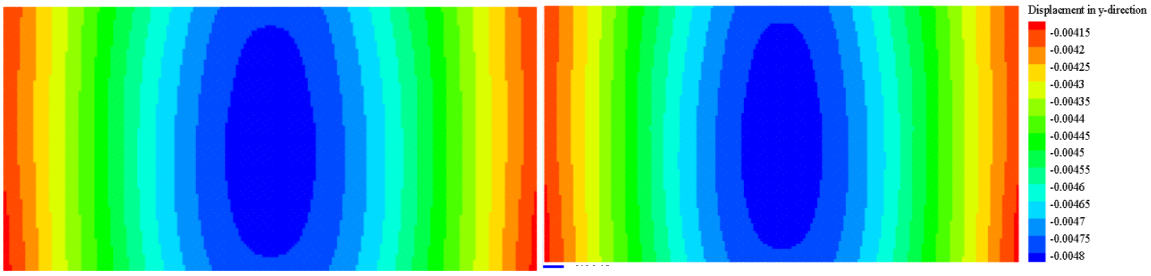


Figure 7. Rigid cylinder impacting an ice plate without damage.



(**a**) (**b**)

Figure 8. Comparison of displacement of ice plate in $y-$direction after contacting: (**a**) original PD method, (**b**) particle−pair method.

In terms of the results, Figure 8 illustrates that the particle−pair algorithm exhibits a similar performance to the original PD algorithm. Subsequently, we delved into the computational efficiency improvement of this method. Figure 9 compares the time consumption between the particle−pair method and the original PD algorithm for various ice discretizations. The specific configurations of the cases with different numbers of ice particles are provided in Table 3. The number of particles ranges from 5000 to 1,440,000.

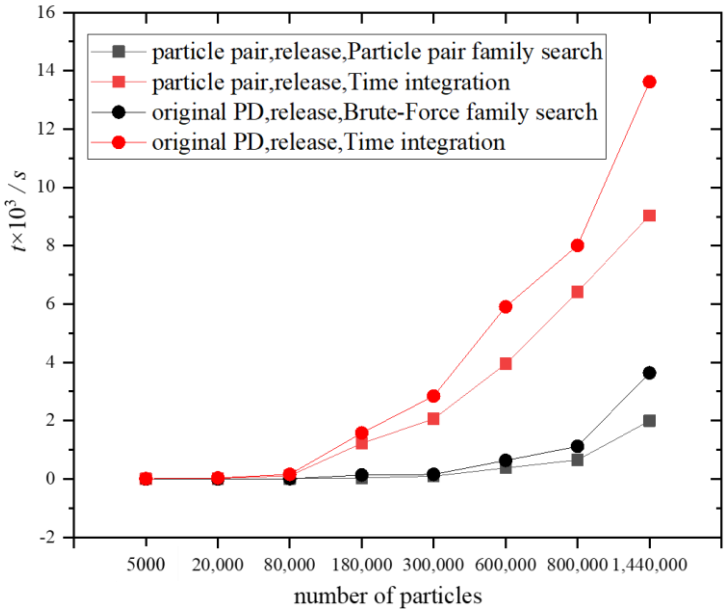


Figure 9. Comparison of calculation time of particle−pair method.

Table 3. Setup of rigid cylinder impacting ice plate by particle−pair method.

Cases	Particle Number	Size of Array in Particle−Pair Method	Size of Particle Array in Original PD Algorithm
case 1	5000	67,318	134,636
case 2	20,000	274,618	549,236
case 3	80,000	1,109,218	2218434
case 4	180,000	9,413,790	18,827,580
case 5	300,000	16,608,234	33,216,468
case 6	600,000	34,594,344	69,188,688
case 7	800,000	46,585,084	93,170,168
case 8	1,440,000	81,529,656	163,059,312

In Table 3, it is observed that the allocated size of the array using the particle−pair method in programming is only half of that in the original PD algorithm. This reduction in memory consumption leads to a significant improvement in computational efficiency.

Figure 9 presents a comparison of the calculation time for both time integration and family member search as a function of the number of particles. It is evident from the comparison curve that the particle−pair method achieves more time savings in both the time integration stage and the particle search stage compared with the traditional PD algorithm. The magnitude of time savings increases as the number of particles increases.

Overall, as the number of particles increases, the time consumption of these two modules (time integration and family member search) increases noticeably. When comparing with the original brute−force search method, the time advantage of the particle−pair algorithm may not be apparent when the number of particles is small (less than 80,000 in Figure 9). However, when the number of particles exceeds 80,000, the particle−pair algorithm exhibits a remarkable reduction in the time consumption of these two parts, saving approximately 78% of the calculation time. In summary, the efficiency improvement achieved with the particle−pair method becomes more significant as the number of particles increases.

3.3. MPI Parallel Technology

In the PD framework, the most commonly used parallel technology is OpenMP, which is based on multithreaded and shared memory parallel mode [34]. OpenMP technology is relatively simple and easy to implement because the calculation domain can be automatically divided into several processors, and it can be easily realized with fewer instructions [35]. However, it still faces limitations regarding the number of particles. On the other hand, message passing interface (MPI) is a widely used cross−language parallel programming technique. The MPI parallel technology enables distributed computation with high scalability, facilitating high−performance parallel computing of clusters and reducing the hardware requirement of a single computer. In this section, the OSB−PD program is compiled based on MPI, enabling the calculation of a large amount of data on hardware with limited memory capacity. The configuration of the computer used for the study is as follows: Intel(R) Core (TM) i7−10875H @ 2.3 GHz, 32 GB RAM, MS Windows 11 × 64.

The computational cost of the PD method primarily lies in the "do loop" of PD force integration on particles and their family members. In other words, the number of particles determines the amount of computation in the algorithm. Therefore, the most effective strategy is to reduce the computational complexity of numerical integration. This can be achieved by employing multiple processors to share the total number of particles, using a domain partition algorithm [36]. The details of the MPI strategy can be found in our previous work [26].

In this section, the case shown in Figure 7 is chosen to study MPI efficiency. The MPI method is applied to conduct the simulation with particles of 1,440,000. The family particle search, integration, and total time under different numbers of thread are calculated. The speedup and efficiency are calculated and shown in Table 4. Speedup is the ratio of the execution time t_s of the serial computation to the execution time $t_p(q)$ of the parallel computation using q processors, $S_p(q) = t_s/t_p(q)$. Efficiency is the ratio of speedup to the number of processor cores used in the parallel program, $E_p(q) = S_p(q)/q$.

In Table 4, the computation time of the original PD algorithm in serial computing mode is about 17,320 s. However, after adopting an MPI parallel strategy, a significant improvement in computing efficiency is observed. In the case with 2 threads, the calculation time is approximately 0.2 times that of the original method. As the number of threads increases, the speedup shows a linear growth trend, reaching 9.46 under 14 threads. However, in terms of computational efficiency, as the number of threads increases, the efficiency gradually reduces. This aspect will be analyzed in detail in the following paragraph. It is evident that the parallel strategy must replace the original calculation method, especially in the case with a large number of particles, as mentioned in Section 2.3.

Table 4. Speedup and efficiency of MPI parallelization with different threads in release mode (unit: s).

Threads	MPI Parallelization			Computing Time of Traditional PD Serial	Speedup	Efficiency
	Search of Family Particles	Time Integration	Total Time			
4	272.234	3319.438	3594.609	17,320.609	4.82	1.20
6	160.531	2316.000	2493.922		6.96	1.16
8	112.797	1877.266	2002.359		8.65	1.08
10	83.484	2284.266	2381.672		7.27	0.727
12	59.828	1988.984	2067.297		8.39	0.698
14	63.343	1758.719	1830.891		9.46	0.676

The time versus the thread number in an MPI scheme for different calculation modules is plotted in Figures 10 and 11. It should be noted that the time consumption of original PD computing is not included in Figure 10, as it has a significantly large value that is impractical to depict in a figure. From Figure 10, it can be observed that the time consumption decreases as the number of threads increases. In Figure 11, it is evident that the calculation achieves a superlinear ratio of when using 8 threads or fewer, that is, $S \geq q$. However, when the number of threads exceeds 8, it transitions to $S < q$. As the number of threads increases, the efficiency exhibits a linear decrease. This is due to the increase in CPU time (lost time) for I/O operations that use memory increases and the increase in data communication as the number of threads increases, as depicted in Figure 12. Moreover, it can be seen that the decrease in time becomes more modest beyond 8 threads.

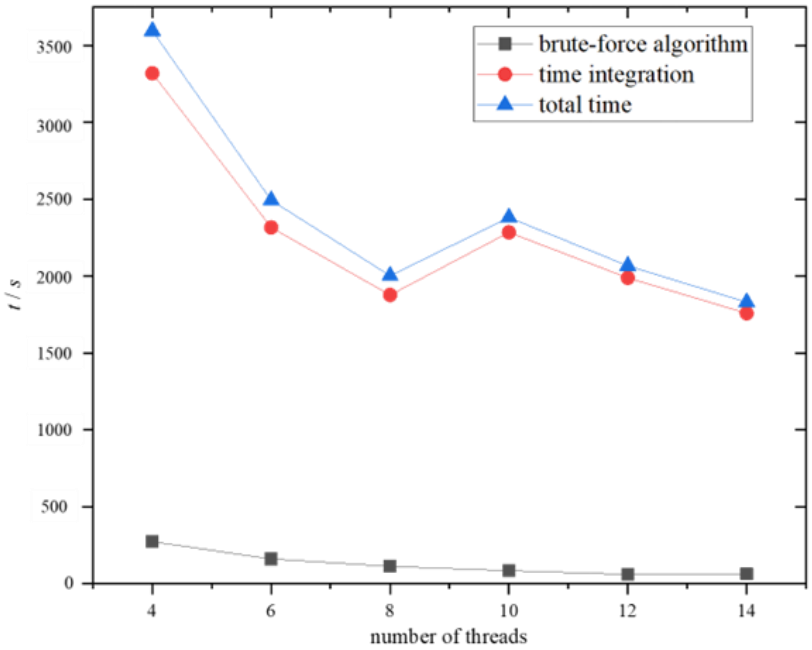


Figure 10. The time versus the thread number in MPI scheme for different calculation modules.

In general, there is a linear relationship between speedup/efficiency and the number of threads. In other words, increasing the number of threads significantly improves the computational efficiency of the algorithm, resulting in an almost linear speedup. However, it is worth noting that the trend differs for 6 and 8 threads compared with other thread counts, particularly with 8 threads. Surprisingly, the speedup of 8 threads is even higher than that of 10 threads. This can be attributed to the fact that computing efficiency is not solely determined by the number of threads but also influenced by data communication, which is determined by domain partitioning. Figure 12 provides insight into this observation, showing that the maximum number of threads communicating simultaneously is 7, 10, and 13 for 6 threads, 8 threads, and 10 threads, respectively.

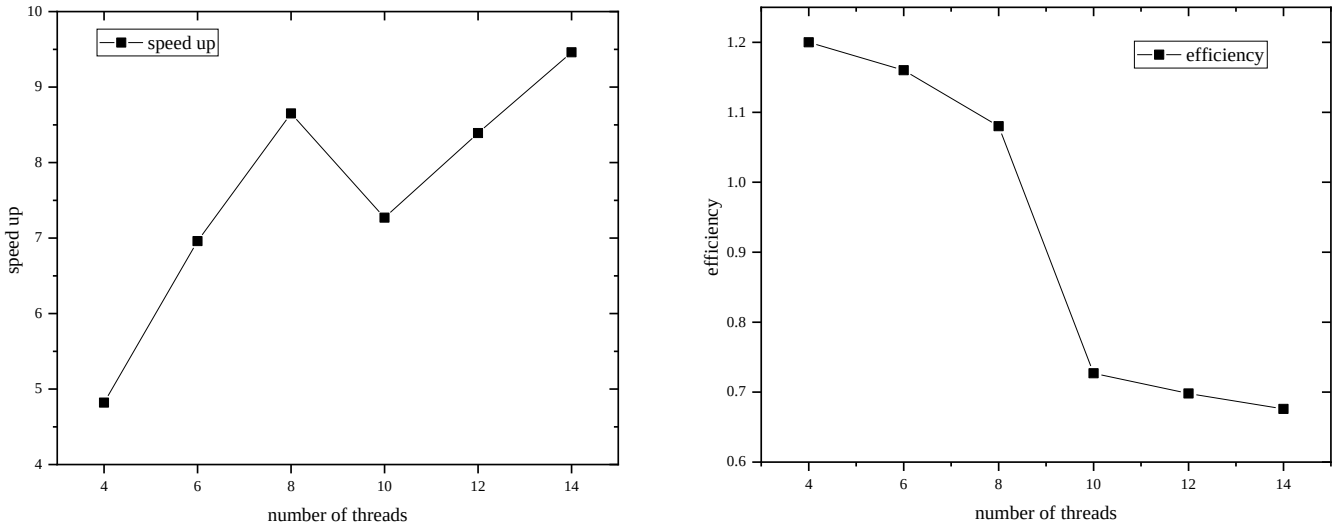


Figure 11. The efficiency of the MPI parallelization.

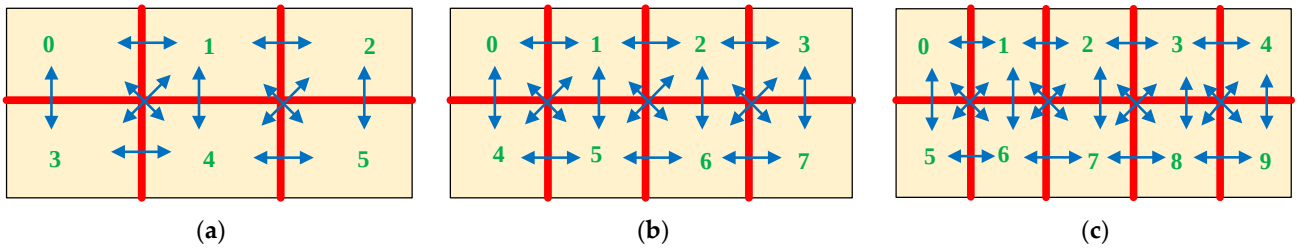


Figure 12. Comparison of data communication in different threads. The yellow plate represents the ice domain, while the thick red lines indicate the thread partitioning of the ice body in parallel computing. The green numbers correspond to the thread numbers (counting from 0), and the blue arrow lines depict the data exchange and direction between different threads ((**a**) 6 threads, (**b**) 8 threads, (**c**) 10 threads).

Additionally, for 8 threads, the communication particles with adjacent threads in the $x-$direction are the same as those in the $y-$direction, both equaling 100. This means that the maximum number of particles communicated between threads simultaneously is 100. However, at 10 threads, the maximum number of communication particles between threads is 100 in the $x-$direction and 80 in the $y-$direction. When the number of particles exchanging information in different directions is unequal, calculation in threads has to wait for other threads to achieve synchronous computing before proceeding. This discrepancy is one of the factors contributing to the lower computational efficiency observed with 10 threads. Furthermore, it should be noted that more threads result in more time loss compared with fewer threads, as MPI implementation on the read/write speed of the storage device (SSD) affects efficiency.

In conclusion, the optimal domain partitioning achieved with 8 threads demonstrates good efficiency. Therefore, it is recommended to prioritize domain partitioning that minimizes the number of communication threads and communication particles in MPI parallel computing. Additionally, it is advisable to ensure that the quantity of particles used for information exchange in all directions is equal or similar.

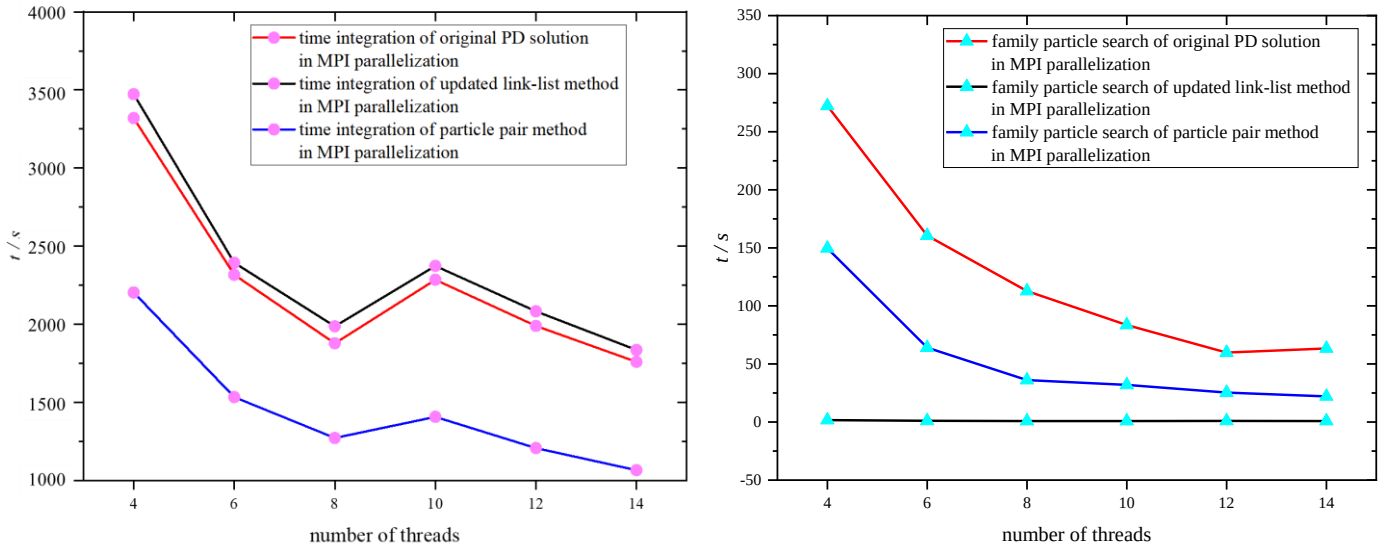
4. Efficiency Analysis of Multimethod Collaborative Computing

In this section, the collaborative computing of multiple methods is investigated in PD programming, building upon the aforementioned three methods. The study further explores the impact of combining multiple methods on computational efficiency and speedup, specifically examining the effects of utilizing the updated link−list method in MPI parallelization and the particle−pair method in MPI parallelization. Additionally,

a comparison of efficiency between MPI parallelization and OpenMP parallelization is conducted. The calculation case used in this section is identical to that of Section 5, involving a rigid cylinder impacting an ice plate.

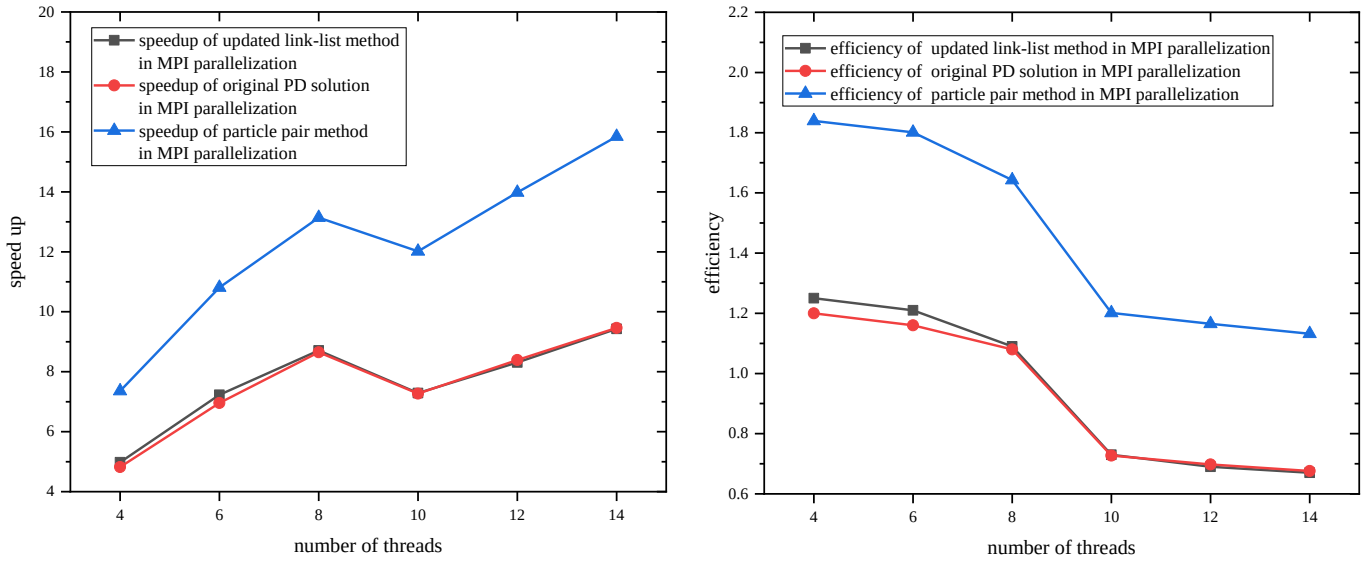
4.1. Updated Link−List Method in MPI Parallelization and Particle−Pair Method in MPI Parallelization

Figure 13 shows a comparison of time consumption for time integration and family particle search among updated link−list method in MPI parallelization, particle−pair method in MPI parallelization, and original PD solution in MPI parallelization. It is evident from the figure that the calculation loop of each modulus significantly decreases with the increase in the number of threads. When the number of threads is equal to 10, the situation is the same as described in Section 3.3, and there is a temporary increase trend. Compared with the original PD method in MPI parallelization, the updated link−list method in MPI parallelization saves much time consumed by family particle search, thereby reducing the total calculation time. However, since the time integration plays a dominant role in the overall program, the efficiency of the updated link−list method in MPI parallelization is less remarkable. The speedup of the two parallel methods aligns with the increase in the number of threads, as depicted in Figure 14a. The updated link−list method in MPI parallelization with 8 threads or fewer is more efficient than the original PD method in MPI parallelization; however, this advantage weakens as the increase in the number of threads, as shown in Figure 14b.



(**a**) (**b**)

Figure 13. Comparison of time consumption among the updated link−list method in MPI parallelization, the particle−pair method in MPI parallelization, and the original PD solution in MPI parallelization. ((**a**) modulus of time integration, (**b**) modulus of family particle search).



(**a**) (**b**)

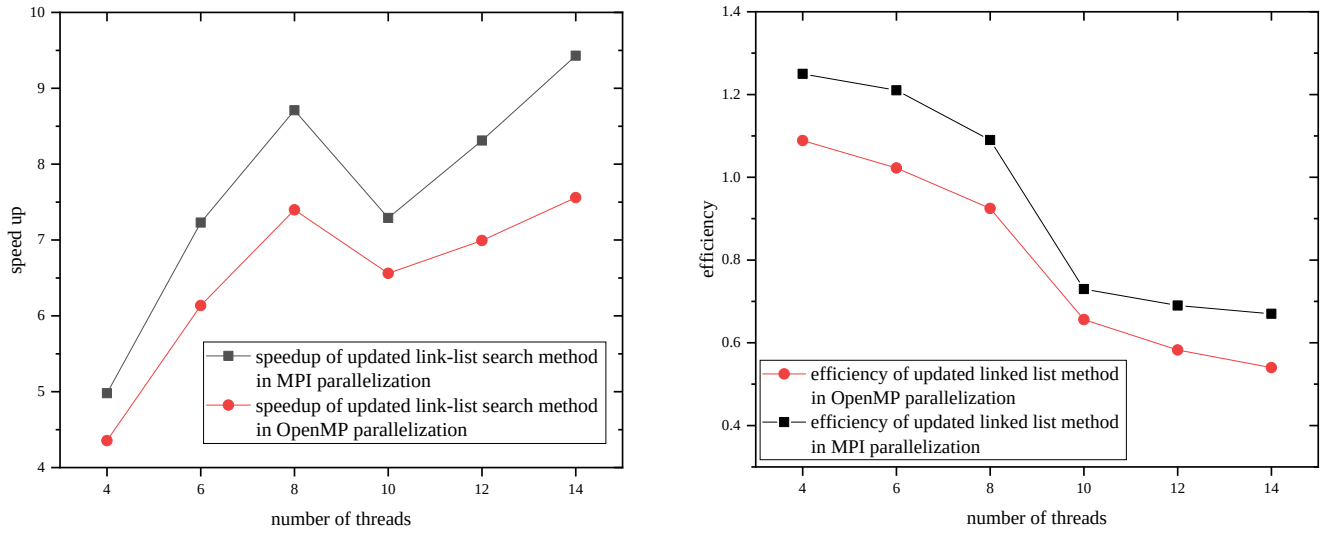
Figure 14. Comparison of (**a**)speedup and (**b**) efficiency among the updated link−list method in MPI parallelization, the particle−pair method in MPI parallelization, and the original PD solution in MPI parallelization.

The particle−pair method in MPI parallelization significantly improves the efficiency of the calculation program, which is about 1.8 times higher than the other two collaborative methods and dramatically reduces the calculation time. With the increase in the number of threads, the speedup of the particle−pair method in MPI parallelization is better than the updated link−list method in MPI parallelization. It can be concluded that the particle−pair method in MPI parallelization plays a significant role in improving the efficiency of PD calculation.

4.2. Comparison between MPI Parallelization and OpenMP Parallelization

In OpenMP parallelization, the brute−force search process cannot be parallelized since the particles in the family member array established during this process have a sequential relationship. Therefore, it is necessary to calculate and assign addresses according to the number of particles in serial computing, which is one of the disadvantages of OpenMP parallelization compared with MPI parallelization. Since the brute−force search process is serial, there is no advantage under the condition of 1,440,000 particles. Therefore, this section directly adopts the updated linked list search method to determine the neighborhood particle information.

As is shown in Figure 15a, the speedup of PD in MPI parallelization is approximately 1.2 times higher than that of PD in OpenMP parallelization, and it remains unchanged regardless of the number of threads. Therefore, it can be concluded that MPI parallelization is a better choice to improve the efficiency of the PD method despite its implementation challenges in the code. In Figure 15b, with the increase in the number of threads, the efficiency of PD gradually decreases, and when the number of threads is 8, the efficiency decreases faster. This trend is the same as that in the summary in Section 3.3. As the number of threads increases, the time benefit resulting from additional parallel threads diminishes due to the growing number of parallel threads.



(**a**) (**b**)

Figure 15. Comparison of the (**a**) speedup and (**b**) efficiency between MPI parallelization and OpenMP parallelization.

5. Engineering Efficiency on Ice–Ship Interaction

In order to demonstrate the application of MPI parallelization in practical engineering cases, an icebreaker breaking level ice was studied in our previous work [25]. In this case, a complete icebreaker model was established to continuously break the level ice, with failure defined by critical stretch. The calculation results validate the accuracy of simulating ice damage. Since our present study primarily focuses on computational efficiency, which was not previously emphasized in our calculations, we will not delve into the details of ice damage and its contour in this section.

In this section, four parallel schemes are set to simulate the process of ship breaking ice. The total time, speed up, and computing efficiency are given in Table 5. The MPI parallel computing efficiency has been greatly improved. As the number of threads increases, the total computation time decreases. Although there is a slight decline in efficiency, computational efficiency can still surpass 90% compared with a single thread, with the calculation speedup increasing linearly. This makes the practical application of MPI parallelization more viable in engineering scenarios.

Table 5. MPI efficiency of the icebreaking process.

Threads	Domain Partitioning	Total Time	Speedup	Efficiency
1	–––––	717,969.6403	–––––	–––––
10	5 × 2	76,280.195	9.89	0.99
20	5 × 4	40,490.68004	18.63	0.93
30	6 × 5	27,785.8704	27.15	0.91
40	8 × 5	13,999.2036	38.89	0.97

Configuration: Intel(R) Xeon(R) E5−2699 v4 @ 2.2 GHz, 128 GB RAM, MS Windows 7 × 64.

6. Discussion

In order to address the issue of low computing efficiency in the original peridynamic (PD) algorithm for simulating ice–structure interaction, this paper adopts three optimization algorithms to simulate a case involving a three−dimensional rigid cylinder impacting an ice plate. These algorithms include the updated link−list method, particle−pair method, and a collaboration method combining MPI parallelization with the aforementioned two methods. Compared with the original algorithms in PD, the influence of the above algorithms on total computing time, speedup, and calculating efficiency is analyzed. Furthermore, the

efficiency of multimethod collaboration is also examined. An engineering case of icebreaker breaking level ice is utilized to demonstrate these methods. The following conclusions can be drawn from the present study:

(1) The updated link−list method significantly improves the calculation efficiency of ice particle search compared with the original PD method. As the number of particles increases, the brute−force method takes much more time in the family member search compared with the updated link−list method.
(2) The particle−pair method reduces both the amount of family particle search and the size of the array storing particles in the horizon. Analysis of the calculation results of ice structures reveals that the particle−pair algorithm improves the computational efficiency by approximately 1.5 to 2 times, and this improvement is independent of the number of particles. Notably, with the increasing number of particles, the time cost of family particle search in the particle−pair algorithm grows slower than that of the original PD algorithm.
(3) With the increase in the number of threads in the MPI scheme, the time consumption of the PD program shows a relatively linear decreasing tendency. At 8 threads and below, parallelism achieves superlinear speedup. However, the number of threads is not the only factor that affects computational efficiency. The number of communicating threads and the number of communicating particles with different domain partitioning also influence efficiency.
(4) The combination methods proposed in the present paper can significantly reduce time consumption. Combined with the numerical simulation results of icebreaker breaking level ice, the particle−pair method in MPI parallelization exhibits the highest efficiency. PD in MPI parallelization outperforms the OpenMP scheme in terms of efficiency.

From our perspective, when dealing with large−scale ice engineering problems. The PD method effectively predicts ice damage. Although parallel computing improves efficiency, it heavily relies on hardware. Based on the research and discussion in this article, we suggest using collaboration between optimized numerical algorithms, such as particle−pair and MPI parallel, to solve the problem of high computational complexity in engineering case analysis.

The method proposed in this paper to enhance the calculation efficiency of ice–structure interaction simulation is only the beginning. For instance, MPI parallelism can also be expanded to high−performance computing clusters to enable a large number of computing nodes. Additionally, if PD can achieve GPU−based parallel computing, it will pave the way for significant advancements in engineering applications. In the future, we will further explore other optimization algorithms to simultaneously enhance efficiency and computational accuracy.

Author Contributions: Conceptualization, Y.Z. and L.Y.; Methodology, Y.Z., L.Y. and K.H.; Validation, Y.Z.; Formal analysis, Y.Z. and S.S.; Investigation, S.S.; Writing—original draft, Y.Z. and G.Z.; Writing—review & editing, L.T. and C.W.; Funding acquisition, C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (Grant No. 52171300).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Silling, S.A. Reformulation of elasticity theory for discontinuities and long-range forces. *J. Mech. Phys. Solids* **2000**, *48*, 175–209. [CrossRef]
2. Silling, S.A.; Askari, E. A meshfree method based on the peridynamic model of solid mechanics. *Comput. Struct.* **2005**, *83*, 1526–1535. [CrossRef]
3. Foster, J.T.; Silling, S.A.; Chen, W.W. Viscoplasticity using peridynamics. *Int. J. Numer. Methods Eng.* **2010**, *81*, 1242–1258. [CrossRef]
4. Silling, S.A.; Epton, M.; Weckner, O.; Xu, J.; Askari, E. Peridynamic States and Constitutive Modeling. *J. Elast.* **2007**, *88*, 151–184. [CrossRef]
5. Nikolaev, P.; Sedighi, M.; Jivkov, A.P.; Margetts, L. Analysis of heat transfer and water flow with phase change in saturated porous media by bond-based peridynamics. *Int. J. Heat Mass Transf.* **2022**, *185*, 122327. [CrossRef]
6. Katiyar, A.; Agrawal, S.; Ouchi, H.; Seleson, P.; Foster, J.T.; Sharma, M.M. A general peridynamics model for multiphase transport of non-Newtonian compressible fluids in porous media. *J. Comput. Phys.* **2020**, *402*, 109075. [CrossRef]
7. Madenci, E.; Roy, P.; Behera, D. *Advances in Peridynamics*; Springer: Berlin/Heidelberg, Germany, 2022.
8. Wang, Q.; Wang, Y.; Zan, Y.F.; Lu, W.; Bai, X.L.; Guo, J. Peridynamics simulation of the fragmentation of ice cover by blast loads of an underwater explosion. *J. Mar. Sci. Tech.* **2018**, *23*, 52–66. [CrossRef]
9. Zhang, Y.; Tao, L.; Wang, C.; Sun, S. Peridynamic analysis of ice fragmentation under explosive loading on varied fracture toughness of ice with fully coupled thermomechanics. *J. Fluids Struct.* **2022**, *112*, 103594. [CrossRef]
10. Vazic, B.; Oterkus, E.; Oterkus, S. In-Plane and Out-of Plane Failure of an Ice Sheet using Peridynamics. *J. Mech.* **2020**, *36*, 265–271. [CrossRef]
11. Li, J.; Wang, C.; Wang, Q.; Zhang, Y.; Jing, C.; Han, D. Peridynamic modeling of polycrystalline S2 ice and its applications. *Eng. Fract. Mech.* **2023**, *277*, 108941. [CrossRef]
12. Lu, W.; Wang, Q.; Jia, B.; Shi, L. Simulation of Ice-Sloping Structure Interactions With Peridynamic Method. In Proceedings of the 28th International Ocean and Polar Engineering Conference, Sapporo, Japan, 10–15 June 2018.
13. Jia, B.; Ju, L.; Wang, Q. Numerical Simulation of Dynamic Interaction Between Ice and Wide Vertical Structure Based on Peridynamics. *Comput. Model. Eng. Sci.* **2019**, *121*, 501–522. [CrossRef]
14. Song, Y.; Liu, R.; Li, S.; Kang, Z.; Zhang, F. Peridynamic modeling and simulation of coupled thermomechanical removal of ice from frozen structures. *Meccanica* **2019**, *55*, 961–976. [CrossRef]
15. Liu, M.H.; Wang, Q.; Lu, W. Peridynamic simulation of brittle-ice crushed by a vertical structure. *Int. J. Nav. Archit. Ocean. Eng.* **2017**, *9*, 209–218. [CrossRef]
16. Liu, R.W.; Xue, Y.Z.; Lu, X.K.; Cheng, W.X. Simulation of ship navigation in ice rubble based on peridynamics. *Ocean Eng.* **2018**, *148*, 286–298. [CrossRef]
17. Vazic, B.; Oterkus, E.; Oterkus, S. Peridynamic approach for modelling ice-structure interactions. In *Trends in the Analysis and Design of Marine Structures*; CRC Press: Boca Raton, FL, USA, 2019.
18. Ye, L.Y.; Guo, C.Y.; Wang, C.; Wang, C.H.; Chang, X. Peridynamic solution for submarine surfacing through ice. *Ships Offshore Struct.* **2020**, *15*, 535–549. [CrossRef]
19. Ye, L.Y.; Wang, C.; Chang, X.; Zhang, H.Y. Propeller-ice contact modeling with peridynamics. *Ocean Eng.* **2017**, *139*, 54–64. [CrossRef]
20. Liu, R.; Xue, Y.; Lu, X. Coupling of Finite Element Method and Peridynamics to Simulate Ship-Ice Interaction. *J. Mar. Sci. Eng.* **2023**, *11*, 481. [CrossRef]
21. Diyaroglu, C. *Peridynamics and Its Applications in Marine Structures*; University of Strathclyde: Glasgow, UK, 2016.
22. Vazic, B. *Multi-Scale Modelling of Ice-Structure Interactions*; University of Strathclyde: Glasgow, UK, 2020.
23. Domínguez, J.M.; Crespo, A.J.C.; Gómez-Gesteira, M.; Marongiu, J.C. Neighbour lists in smoothed particle hydrodynamics. *Int. J. Numer. Methods Fluids* **2011**, *67*, 2026–2042. [CrossRef]
24. Guo, C.; Han, K.; Wang, C.; Ye, L.; Wang, Z. Numerical modelling of the dynamic ice-milling process and structural response of a propeller blade profile with state-based peridynamics. *Ocean. Eng.* **2022**, *264*, 112457. [CrossRef]
25. Yuan, Z.; Longbin, T.; Chao, W.; Liyu, Y.; Chunyu, G. Numerical study on dynamic icebreaking process of an icebreaker by ordinary state-based peridynamics and continuous contact detection algorithm. *Ocean Eng.* **2021**, *233*, 109148. [CrossRef]
26. Zhang, Y.; Tao, L.; Wang, C.; Ye, L.; Sun, S. Numerical study of icebreaking process with two different bow shapes based on developed particle method in parallel scheme. *Appl. Ocean. Res.* **2021**, *114*, 102777. [CrossRef]
27. Madenci, E.; Oterkus, E. *Peridynamic Theory and Its Applications*; Springer: New York, NY, USA, 2014. [CrossRef]
28. Silling, S.A.; Askari, E. Peridynamic modeling of impact damage. In Proceedings of the ASME/JSME 2004 Pressure Vessels and Piping Conference, San Diego, CA, USA, 25–29 July 2004; pp. 197–205.
29. Zhang, Y.; Tao, L.; Ye, L.; Wang, C.; Sun, S.; Lu, W. An updated fast continuous contact detection algorithm and its implementation in case study of ice-structure interaction by peridynamics. *Mar. Struct.* **2023**, *89*, 103406. [CrossRef]
30. Bobaru, F.; Foster, J.T.; Geubelle, P.H.; Silling, S.A. *Handbook of Peridynamic Modeling*; CRC Press: Boca Raton, FL, USA, 2016. [CrossRef]
31. Liu, G.-R.; Liu, M.B. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*; World Scientific: Singapore, 2003.

32. Zhang, Y. Numerical Study of Ice and Ice-Ship Interaction Based on Ordinary State-Based Peridynamics. Ph.D. Thesis, Harbin Engineering University, Harbin, China, 2022. (Unpublished Doctoral Dissertation). (In Chinese)
33. Han, K. A particle pair method to improve PD computation efficiency and its MPI parallel strategy. *J. Harbin Eng. Univ.* **2022**, *accepted*.
34. Prakash, N.; Stewart, R.J. A Multi-threaded Method to Assemble a Sparse Stiffness Matrix for Quasi-static Solutions of Linearized Bond-Based Peridynamics. *J. Peridynamics Nonlocal Model.* **2021**, *3*, 113–147. [CrossRef]
35. Chambon, G.; Bouvarel, R.; Laigle, D.; Naaim, M. Numerical simulations of granular free-surface flows using smoothed particle hydrodynamics. *J. Non-Newton. Fluid Mech.* **2011**, *166*, 698–712. [CrossRef]
36. Cui, X.D.; Habashi, W.G.; Casseau, V. MPI Parallelisation of 3D Multiphase Smoothed Particle Hydrodynamics. *Int. J. Comput. Fluid. D* **2020**, *34*, 610–621. [CrossRef]