

Article

A Fast Time Synchronization Method for Large Scale LEO Satellite Networks Based on a Bionic Algorithm

Yue Xu ¹, Tao Dong ¹, Jie Yin ^{1,*}, Ziyong Zhang ¹, Zhihui Liu ¹, Hao Jiang ² and Jing Wu ²

¹ State Key Laboratory of Space-Ground Integrated Information Technology, Space Star Technology Co., Ltd., Beijing 100095, China; xubancun509@163.com (Y.X.); dongtaoandy@163.com (T.D.); zhangkeke0831@126.com (Z.Z.); liuzhihui1225@163.com (Z.L.)

² School of Electronic Information, Wuhan University, Wuhan 430072, China; jh@whu.edu.cn (H.J.); wujing@whu.edu.cn (J.W.)

* Correspondence: kingjack33333@126.com

Abstract: A fast time synchronization method for large-scale LEO satellite networks based on a bionic algorithm is proposed. Because the inter-satellite links are continuously established and interrupted due to the relative motion of the satellites, the topology of the LEO satellite networks is time varying. Firstly, according to the ephemeris information in navigation messages, a connection table which records the connections between satellites is generated. Then, based on the connection table, the current satellite network topology is calculated and generated. Furthermore, a bionic algorithm is used to select some satellites as time source nodes and calculate the hierarchy of the clock transmission tree. By taking the minimum level of the time transmission tree as the optimization objective, the time source nodes and the clock stratum of the whole satellite networks are obtained. Finally, the onboard computational center broadcasts the time layer table to all the satellites in the LEO satellite networks and the time synchronization links can be established or recovered fast.

Keywords: LEO satellite networks; time synchronization; time source selection; particle swarm optimization



Citation: Xu, Y.; Dong, T.; Yin, J.; Zhang, Z.; Liu, Z.; Jiang, H.; Wu, J. A Fast Time Synchronization Method for Large Scale LEO Satellite Networks Based on a Bionic Algorithm. *Photonics* **2024**, *11*, 475. <https://doi.org/10.3390/photonics11050475>

Received: 23 November 2023

Revised: 2 February 2024

Accepted: 16 February 2024

Published: 19 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The construction of large-scale satellite communication constellations is developing rapidly, and time-sensitive services play an important role in such fields as finance, telemedicine and intelligent transportation [1,2], as shown in Figure 1. Due to propagation delay and the Doppler effect, satellite communication requires time and frequency synchronization [3]. Satellite time synchronization is a basic condition for conducting inter-satellite time-sensitive services, which directly affects parameters such as bandwidth, delay, and jitter [4–7]. Time-sensitive services between satellites rely on long-term continuous and reliable time synchronization sources.

Satellite time synchronization methods include star-ground two-way time synchronization [8], inter-satellite time synchronization based on two-way ranging and time transmit (TWRTT) technology [9], and time synchronization using navigation signals [10–12]. The time synchronization method using navigation constellations has few error sources and high time synchronization accuracy, and it is an important process for time synchronization in satellite constellations.

However, in low earth orbit (LEO) satellite networks, inter-satellite links are difficult to maintain due to the drastic changes in relative motion speed between satellites, and links are constantly established and interrupted [13–15]. Meanwhile, when affected by interference or space environment factors, such as by solar flares and atmospheric gravity wave disturbances [16,17], the ionosphere will be perturbed, which may lead to distortions and the mis-coding of the signals received by the receivers, which cannot be decoded [18]. These can lead to the network topology being time-varying. In addition, time transfer

cannot be accomplished when the time synchronization source fails or the time transmission link fails. Therefore, it is necessary to quickly reselect the clock source and calculate the time synchronization transmission path according to the current network topology in order to achieve the continuous and stable transmission of real-time services.

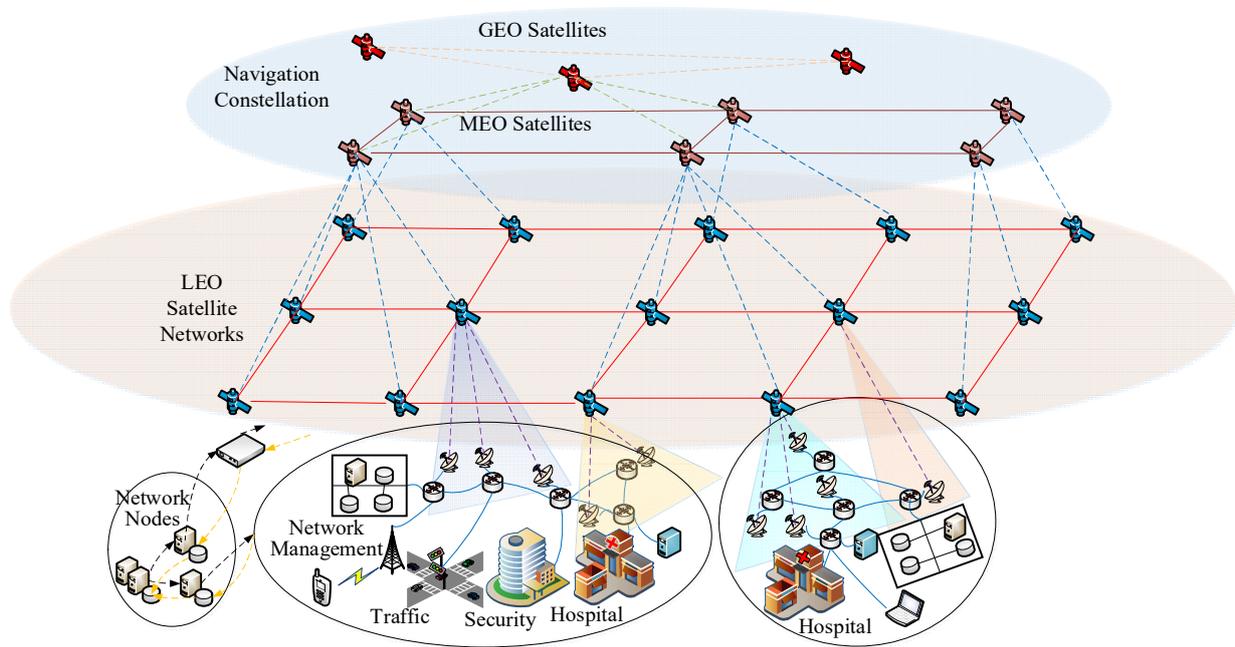


Figure 1. Schematic diagram of space and earth integration networks.

On the Internet, clock information is transmitted from the clock source to the next node level by level, and each node level is a clock level [19,20]. Every level of clock information transmission will generate errors. As the number of times clock information is transmitted increases, the error will become larger, and the clock synchronization performance will deteriorate. Therefore, the clock synchronization performance is closely related to the clock hierarchy in the network, and the fewer clock hierarchies, the better the clock synchronization performance. Therefore, during the algorithm iteration in this article, the clock hierarchy is established based on the clock transfer tree, and the clock hierarchy of all satellites is weighted and summed. The minimum value is taken as the optimization objective, and the clock synchronization performance is best when the clock hierarchy value is the smallest.

The traditional traversal algorithm calculates the clock delivery path and clock hierarchy by using each satellite as a clock source in turn [15,21], which requires all possible cases to be calculated sequentially, and the optimal result can only be determined by comparing the results of the calculations for all cases [22]. Therefore, as the number of satellites in the network increases, the amount of computation increases dramatically and can even be exponential. Facing the LEO satellite network with time-varying topology under normal circumstances, although the traversal algorithm can pre-calculate the lookup table, when there are unforeseen circumstances that lead to the failure of the satellite link and the need to re-calculate the clock transfer path, the traversal algorithm has a large amount of computational workload, and it cannot compute the results in time, which generates unpredictable delay and jitter, greatly affecting the transmission and processing of information.

The proposed method selects multiple time reference satellites using the bionic algorithm based on the change in LEO satellite network topology with the constraint of minimizing the sum of weighted clock tiers. The bionic algorithm calculates by iteration, and at each iteration the simulation result is closer to the optimal result, instead of calculating all the possible scenarios as the traversal algorithm does, so that consequently a large number of unnecessary calculations are saved; the time of calculation is greatly

reduced compared with the traversal algorithm, which can quickly calculate the optimal clock transfer path for the whole network and achieve the best time synchronization effect. When a single or part of the interstellar link fails due to unexpected circumstances, the topology of the satellite network changes, and the clock transmission path needs to be recalculated. The iterative approach of the bionic algorithm is far more efficient than the traversal algorithm that calculates one by one. In this paper, for the same model, the traversal algorithm needs to compute 11^6 times and takes 195.63 s, while the proposed bionic algorithm, with a minimum of only 10 iterations and 0.043 s, can enable the time synchronization network to recover quickly, which is an important application prospect in LEO satellite networks.

2. Time Synchronization Scheme

2.1. Clock Transfer Trees Construction

In the method, all the LEO satellites are set to move back to the initial latitude at the beginning of a complete operating cycle, and the time interval between two consecutive changes in the LEO satellite network topology are designated as a time slice [21]. For each time slice, the satellite is set to establish inter-satellite links with the adjacent satellites in the same orbit on the front and back sides and two adjacent satellites located in adjacent orbits on the left and right sides.

For polar orbit satellites, at different latitude positions, the fast relative motion between the adjacent orbit satellites causes periodical establishments and disconnections of inter-satellite links. When the satellites move to the high latitude region, the satellites in the two neighboring orbits have large relative movement speeds. It is difficult to maintain the inter-satellite links, and even causes the links to be disconnected. Thus, in our approach, it is set so that when the LEO satellites move to a high latitude region that is larger than a 70° south and north latitude, the inter-satellite links in adjacent orbit are put into disconnected states and no connections exist between the satellites in the network topology; otherwise, the satellites links will remain connected. Several satellites are set as time sources in each time slice to achieve a flat time synchronization hierarchy structure and improve clock synchronization accuracy. The latitude position of each satellite can be obtained based on navigation messages. The inter-satellite link statuses and network topology of the time slice are determined according to the latitude positions in the proposed means.

For the traversal algorithm and the proposed method, the clock transfer trees are constructed in the same way. We use 1–66 as the number for 66 satellites, with the first orbital plane numbered 1–11, the second orbital plane numbered 12–22, and so on. The time transfer trees are established based on their own time source satellites as root nodes with the clock layers of 0 [23].

Firstly, the time source satellites are marked as visited to avoid loops, and the smallest numbered time source is used as the first root node. Then, the satellites, which have connections with the first root node, are designated as the time cascade level 1 and identified as visited. Meanwhile, their parent nodes of stratum 0 are recorded. Then, the second smallest numbered time source satellite is defined as the second root node with the clock layer of level 0, too. The satellites connected to the second root node, which are not identified, are set as the time hierarchy of stratum 1. The nodes are also marked as visited with their parent nodes recorded, too. In this way, all satellites with clock stratum layer 1 can be found and marked as visited, and their parent nodes are recorded.

Secondly, the satellites connected to the smallest numbered satellite with time level 1 in the first clock transfer tree are found. After removing the already identified satellites, the chosen satellites are set to be the time stratum 2 and marked as visited, with their parent nodes recorded. Following the same measure, all the satellites connected to the other satellites with clock layer 1 can be found. After removing the already identified satellites, the found satellites are set to time level 2, and marked as visited, with their parent nodes recorded. Then, using this method, all satellites clock layers and parent nodes can be determined.

Finally, time transfer trees, which have the same number of the time source satellites, are constructed based on the parent nodes recorded by each satellite.

2.2. Sum of Weighted Clock Hierarchy Calculation

The weight value of the satellite is the same as the time stratum of itself. For example, the weight value of the time source satellite of the 0 clock layer is 0. The weight values of all satellites are summed to calculate the weighted time level, and the minimum value of the sum of the weighted clock stratum is considered as the optimization target. Using the bionic optimization algorithm, time sources are initialized and randomly selected with the number of D . The time layer of each satellite is estimated according to the above measurement, and then the sum of the weighted clock levels is obtained. The D time sources are consistently selected iteratively until the minimum sum of the weighted time cascades is obtained through optimization. At this time, when the D time sources are the optimal solution, the clock stratum and the parent node of each satellite are recorded. To complete the setting of time hierarchy of all satellites in the LEO satellite networks, the synchronization information is broadcasted to other satellites.

In this paper, the particle swarm optimization (PSO) algorithm is used [24–28], which is one kind of the bionic algorithms. The low-orbit constellation takes the Iridium constellation as an example, which has 66 low-orbit satellites divided into six orbital planes. For the Iridium constellation, six satellites are selected as the time references, as every plane has one time source. The traditional traversal approach for obtaining the six time references with the smallest weighted clock cascade needs to calculate all combinations of time references, i.e., 11^6 , in order to determine the optimal combination of time references. In contrast, the proposed means, with the number of particles set to 20 in the PSO algorithm, only requires a minimum of 10 iterations to determine the optimal time source combination.

$$s_k = \sum_{i=1}^N a_{i,k} \quad (1)$$

In Equation (1), s_k represents the sum of the weighted time stratum of the k -th clock source satellite combination in the time slice, N represents the number of all the satellites in the constellation; in this article, it is 66. $a_{i,k}$ represents the time hierarchy of the i -th satellite in the k -th clock source combination in the time slice.

In order to obtain a better global time synchronization effect, the sum of the clock weighted layer should be as small as possible, so as to minimize the time level of each satellite. There are 11 selections of clock sources for each orbital plane with 11 satellites, and a total of 11^6 combinations for six orbital planes, namely 1,771,561.

In order to make a comparison with the method proposed in this paper, the time layers calculations have also been performed with the traditional traversal method [29,30]. The traditional traversal measure mainly differs from the PSO algorithm in the time source selection, while the process of calculating the clock transfer trees and the sum of the satellite weighted time cascade is the same as described earlier. Traditional traversal methods calculate the weighted clock stratum for each combination in each time slice. That is, for the first time slice, $s_1-s_{1771561}$ are calculated. Firstly, six time sources in six orbital planes are selected, such as the first combination of the satellites with the numbers of 1, 12, 23, 34, 45, 56. Secondly, the connection relationship between satellites is obtained based on the satellite network topology of the time slice, and the clock transfer trees are obtained based on this connection relationship and the time source satellite number. Finally, s_k is calculated according to the clock transfer trees.

3. Results

The traditional traversal approach is shown in Algorithm A1. All combinations of time sources are selected with the increasing number of orbital planes and satellites, and for each combination, the sum of the weighted clock layers is computed.

All 1,771,561 time source combinations are calculated according to the above steps, and a minimum sum of the weighted clock hierarchy of 102 is obtained. The optimal time combination serial numbers indicate the positions in the above 1,771,561 combinations. A total of eight combinations can achieve this goal, and the corresponding satellite numbers for each combination are shown in Table 1. The program is set up to monitor the time and it took a total of 195.63 s to calculate all 1,771,561 clock source combinations.

Table 1. Optimal clock source combination table of the first slice obtained by traversal algorithm.

Group	Optimal Clock Source Combination Serial Numbers	Satellite Numbers						
1	993,418	7	13	32	38	45	63	
2	993,429	7	13	32	38	46	63	
3	1,133,788	8	12	27	43	46	62	
4	1,148,429	8	13	27	43	46	62	
5	1,510,256	10	16	24	41	49	66	
6	1,524,897	10	17	24	41	49	66	
7	1,678,566	11	16	30	35	49	65	
8	1,678,577	11	16	30	35	50	65	

The time transfer trees corresponding to the group 2 and group 5 time source combinations of Table 1 are shown in Figure 2a,b, respectively, in which the time sources are used as the root nodes of the clock transfer trees. The numbers in the time transfer trees denote the numbers of the satellites, and the column of serial numbers 0–3 on the left side of the clock transfer trees denotes the time cascade corresponding to the satellites in this row. The time synchronization information is transferred from the time source satellites to the other satellites along the clock hierarchy with the connections between the satellites.

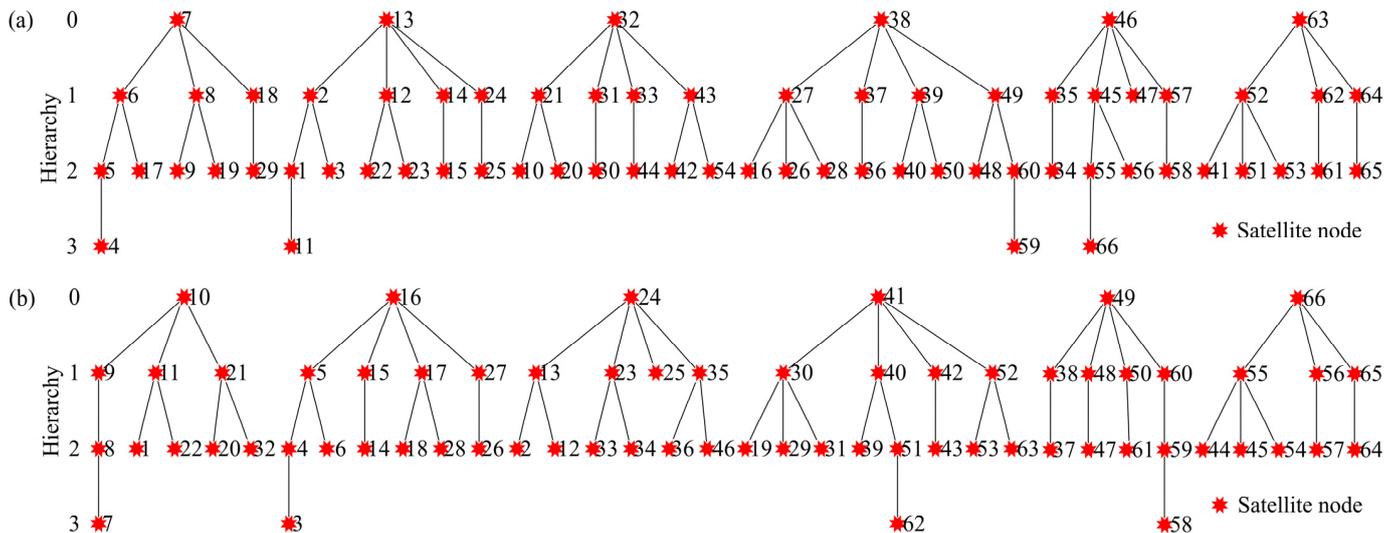


Figure 2. Time transfer trees: (a) the second group, (b) the fifth group in Table 1.

The minimum sum of the weighted cascades of the time slice is $\min\{s_1, s_2, \dots, s_{1,771,561}\}$, and its corresponding optimal time source combination can be found with the *find* function in Algorithm A1, whose searching condition is $s_k == \min\{s_1, s_2, \dots, s_{1,771,561}\}$. The minimum sum of weighted clock level calculated by the means of the traversal is 102.

In the particle swarm algorithm of this thesis, the size of the number of population particles NP is generally between a few and in the tens. When the NP is small, the iteration can easily fall into the local optimal solution. When the NP is large, the results of the calculation of many particles in each iteration are not facilitated, which wastes the computational resources and reduces the efficiency, so the NP can be set to 20. N is the total

number of satellites of the Iridium constellation, which is set to 66. The particle dimension D is the time base number, which is set to six. SN is the number of satellites in each orbital plane, which is set to 11. The maximum number of iterations is set based on the iteration results. If a small number of iterations can produce optimal iteration results, the maximum number of iterations can be set to a very small number of iterations, so as to avoid useless calculations. According to the results of the PSO calculations in this example, usually dozens of iterations can produce the optimal results, thus T is set to 100. The maximum number of iterations affects the inertia weight ω of the particles as shown in Equation (2). As shown in Equation (5), the learning factors affect the motion speed of the particles, which cannot be too large compared to the weight ω . The two learning factors $c1$ and $c2$ are usually set to 1.5 in the PSO algorithm. As the near-polar orbit LEO satellites have a high relative speed of motion at high latitudes, inter-satellite links are difficult to maintain, so the inter-satellite link is set to be disconnected when the satellite's latitude is 70° . Because there are 11 satellites in each orbital plane, in order to facilitate the program calculation for each orbital plane, the time base number maximum $Xmax$ is 12 and the minimum $Xmin$ is 1 so that in the use of Equation (3) calculations the clock source satellite number combination $x1$ can be assigned to the number of all the numbers of 1–11. Since the time reference number is an integer, it is rounded to obtain the initialized time reference x (value range 1–11). The particle motion speed is used to control the magnitude of the change in the clock source satellite number for each update, because there are only 11 satellites in each orbital plane, and the satellite number value cannot grow too much in each iteration, otherwise it may exceed the value range of the number 1–11, or skip the optimal value in the middle. So, the maximum value of $Vmax$ is set to 2, and the minimum value of $Vmin$ is set to 0, i.e., the satellite number is increased by 1 or 2 in each iteration. The above variables are calculated by the following formulas:

$$\omega = (0: 1/(T - 1): 1) \times (0.4 - 0.9) + 0.9 \tag{2}$$

$$x1 = \mathbf{rand}(NP, D) \times (Xmax - Xmin) + Xmin \tag{3}$$

$$x = \mathbf{floor}(x1) \tag{4}$$

$$v_j = \omega(l) \times v_j + c1 \times \mathbf{rand} \times (pbest_j - x_j) + c2 \times \mathbf{rand} \times (gbest - x_j) \tag{5}$$

$$x1_j = x_j + v_j \tag{6}$$

where the *rand* function returns a random number uniformly distributed in the interval (0, 1), the *floor* function rounds an element to the nearest integer less than or equal to the element, j denotes the particle number, l denotes the number of loop iterations, v is the particle motion velocity, *pbest* is the historical optimal solution for each particle, and *gbest* is the historical optimal solution for all particles and the optimal time source combination. The main initial parameters and their corresponding values in the PSO algorithm are shown in Table 2.

The optimization objective of the PSO is generated through the *func1* function, which returns a numerical value of *sumstr*, as shown in Equation (7).

$$sumstr = \mathbf{func1}(x, A, N, D) \tag{7}$$

The *func1* function shown in Algorithm A2 is used to compute one optimization per particle, where x represents the six time source numbers selected by a particle, A represents the connection table with the topology of the LEO satellite networks in this time slice, N is the total number of satellites in the LEO satellite networks, and D is the number of time source numbers. For particle j , the return value *sumstr* of function *func1* is calculated by Equation (8) in function *func1* and a_i is the time hierarchy of the i -th satellite.

$$sumstr = \sum_{i=1}^N a_i \tag{8}$$

$$pbest_j = \begin{cases} sumstr, & pbest_j > sumstr \\ pbest_j, & pbest_j < sumstr \end{cases} \tag{9}$$

$$gbest = \begin{cases} gbest, & pbest_j > gbest \\ pbest_j, & pbest_j < gbest \end{cases} \tag{10}$$

Table 2. Parameters table.

Parameters	Value
Total number of satellites (<i>N</i>)	66
Number of time sources (<i>D</i>)	6
Satellites in each orbital plane (<i>SN</i>)	11
The latitude at which the inter-satellite link is set to be disconnected	70°
Number of particles (<i>NP</i>)	20
Maximum number of iterations (<i>T</i>)	100
Learning factors (<i>c1, c2</i>)	1.5
Maximum value of time source number for each orbital plane (<i>Xmax</i>)	12
Minimum value of time source number for each orbital plane (<i>Xmin</i>)	1
Initialization time source (<i>x</i>)	[1, 11]
Maximum particle velocity (<i>Vmax</i>)	2
Minimum particle velocity (<i>Vmin</i>)	0

Whenever *sumstr* generates a new value, the particle is compared with its own historical optimal value *pbest_j* by Equation (9). If *sumstr* is smaller, the particle updates and records its own optimal time sources *x_j* and the minimum sum of weighted time layers *pbest_j*. Otherwise, it does not record and update. Then, if *pbest_j* is smaller than the global minimum sum of the weighted clock cascades *gbest*, the global minimum sum of weighted time stratum and global optimal time sources are updated by Equation (10). Otherwise, it does not record and update. Finally, the particle updates its own speed *v_j* and time source satellite numbers *x1_j*; and an iteration of this particle is completed. After all iterations of all particles, the clock transfer trees are generated using the global optimal time source combinations. The main algorithm is shown in Algorithm A3.

The minimum sum of weighted time stratum obtained after 10 iterations is 102, as shown in Figure 3. The obtained time source combinations are the same as the second group in Table 1. Six clock transfer trees are constructed with these six time sources as the root nodes. The LEO satellite network topology and clock delivery path are schematically shown in Figure 4. Figure 4 illustrates the network topology for the first time slice and the clock delivery paths established based on the clock delivery trees in Figure 2a. The clock source satellites are marked in yellow, while the other satellites are marked in blue. The numbers represent the other satellites' numbers, and the arrow indicates the clock delivery paths.

The curves of the number of iterations of the PSO algorithm and the traversal algorithm with respect to the sum of the minimum weighted clock levels are shown in Figure 3. Using the time monitoring program, the time for 10 iterations of the PSO algorithm is 0.043 s and for 11⁶ iterations of the traversal algorithm is 195.63 s, respectively.

The PSO algorithm has significantly fewer iterations compared to the traversal algorithm. Analyzed in principle, the traversal algorithm needs to compute all possible clock source combination cases sequentially, i.e., for *D* clock sources, each clock source combination case is to add one to only one of the clock source satellite number each time and compute that clock source combination, and the optimal result can only be determined by comparing the size of the sum of the weighted clock hierarchies of all clock source combination cases. These 11⁶ calculations are independent of each other, so the results of each calculation do not benefit the results of the next calculation, which leads to an increase in the number of clock source combinations with the increase in the number of

satellites in the network and a corresponding increase in the amount of calculations in the traversal algorithm.

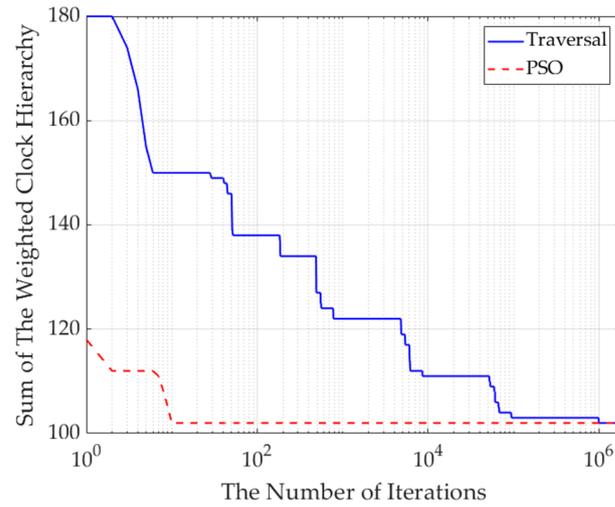


Figure 3. The curves of the number of iterations of the PSO algorithm and the traversal algorithm with respect to the minimum sum of the weighted time in the first time slice.

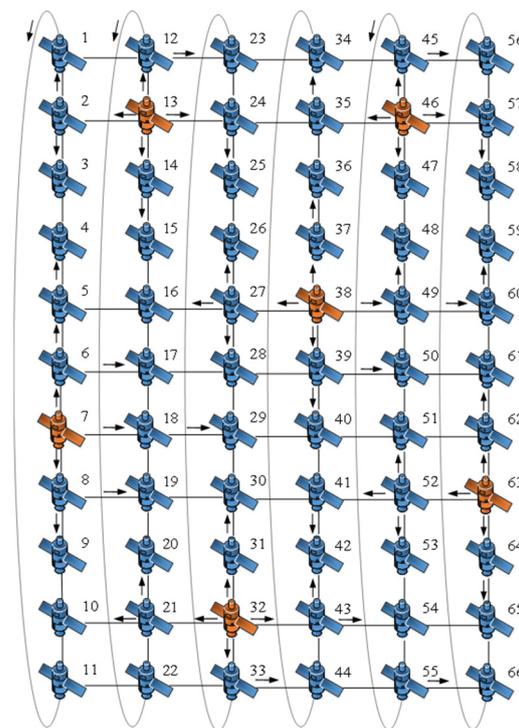


Figure 4. Schematic diagram of clock transmission path and topology of the first slice in LEO satellite networks.

In the PSO algorithm, parameters such as D clock source satellites for NP particles are updated simultaneously each time, and the updated results are used to update the parameters for the next iteration, and each update is beneficial to make the computational results closer to the optimal solution. Each particle represents a kind of time source combination, and it can be seen from Equation (3) that for each particle, PSO updates the D clock source satellite numbers at the same time each time, instead of only one at a time as in the traversal algorithm. Meanwhile, the PSO algorithm utilizes iterative computation, and there is a correlation between the results of the two adjacent computations, as shown

in Equations (4) and (5); the parameters v_j and x_j of the previous iteration are computed to cover the current v_j and x_j , i.e., the parameters of the current iteration are updated using the parameters of the previous iteration, so that the results of the previous computation has a contributing effect on the next computational results. In addition, in Equation (3), PSO updates NP particles at one time simultaneously, i.e., NP kinds of combinations and the computational results of these NP particles update $pbest_j$ and $gbest$, and then update v_j and x_j of each particle according to Equations (4) and (5) so that the iteration of each particle can benefit from the iterative process of other $NP-1$ particles. In summary, instead of enumerating sequentially as the traversal algorithm does, the PSO algorithm significantly improves computational efficiency through these beneficial update mechanisms.

Four time source combinations can be generated after the program has run for several times, as shown in Table 3.

Table 3. Optimal time source combinations obtained by PSO algorithm.

Satellite Number						Iterations
7	13	32	38	46	63	10
10	16	24	41	49	66	62
8	13	27	43	46	62	20
11	16	30	35	49	65	25

In order to study the time synchronization effect of the method in the case of network topology changes, we obtained another time slice of the constellation in one operation cycle by simulation, and the constructed network topology is shown in Figure 5. In the network topology, the numbers represent the satellites' numbers. The difference with the network topology in Figure 4 is that there is no connection between the satellites in row 8 in Figure 5, i.e., there is no inter-satellite link between the satellites in neighboring orbits. The optimal clock source combinations are calculated for the network topology shown in Figure 5 using the traversal algorithm shown in Algorithm A1, and Table 4 lists the optimal clock source combination serial numbers and the corresponding clock source satellite numbers.

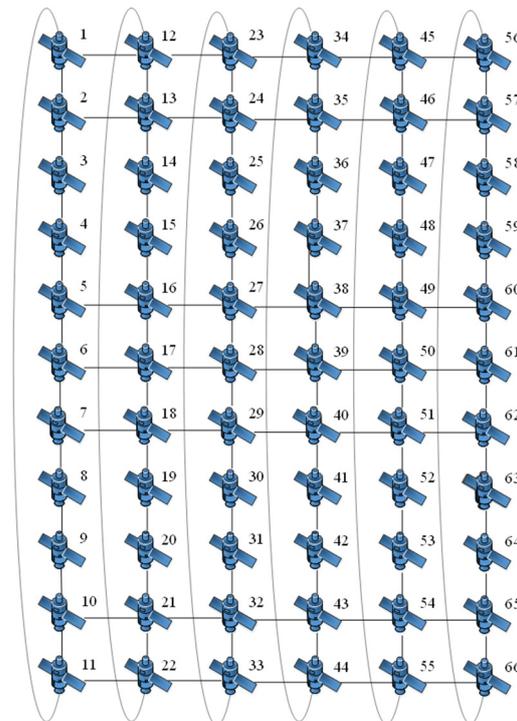


Figure 5. Network topology for another time slice.

Table 4. Optimal clock source combination table.

Group	Optimal Clock Source Combination Serial Numbers	Satellite Numbers					
1	94,277	1	18	27	43	46	62
2	778,085	5	21	24	40	49	66
3	993,477	7	13	32	38	51	56
4	1,677,285	11	16	29	35	54	60

We also calculated the optimal clock source combinations using the PSO algorithm for the network topology shown in Figure 5, obtaining the fourth group clock source combination in Table 4. Clock delivery trees are constructed for the fourth group of clock source combinations, as shown in Figure 6. And, the sum of weighted clock levels is calculated as 103 based on the clock transfer trees in Figure 6. The value is larger than the sum of weighted clock levels 102 in the first time slice, which indicates that the clock synchronization effect is degraded. This is because the network topology has changed in this time slice, the links between satellites have been reduced, the available clock transfer paths have been reduced, and the matrix connection table *A* has been changed, which affects the simulation results according to Equation (7), so the simulation results become worse accordingly.

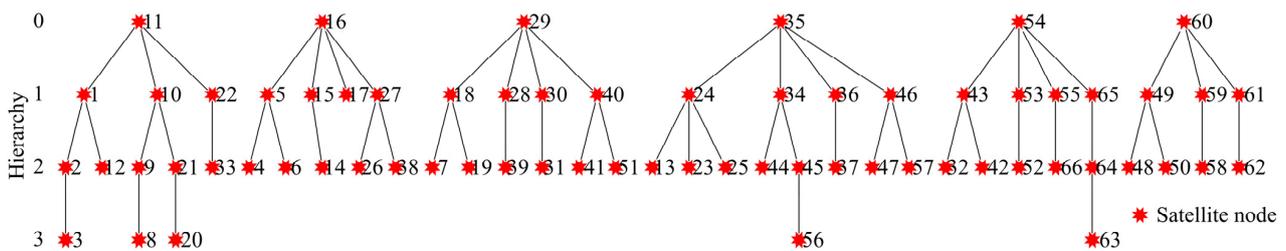


Figure 6. Time transfer trees of the fourth group in Table 4.

The iteration times of the two algorithms are compared in Figure 7. The traversal algorithm iterates to the 94,277th time to obtain the optimal result, but still needs to calculate all 11^6 times to determine, while the PSO algorithm only needs to iterate 43 times to obtain the optimal result. Using a time monitoring program, it takes 190.96 s for the traversal algorithm to calculate 11^6 combinations, and 0.039 s for the PSO algorithm to calculate 43 times.

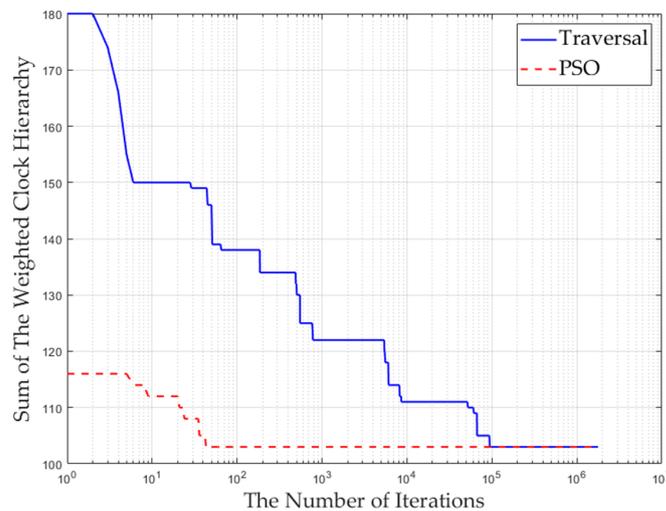


Figure 7. The curves of the number of iterations of the PSO algorithm and the traversal algorithm with respect to the minimum sum of the weighted time in another time slice.

4. Discussion

For all the optimal time source combinations found by the traversal algorithm, the PSO algorithm can only obtain one of the optimal time source combinations each time. For LEO satellite network application scenarios, it is important to find one optimal combination immediately, when the time source satellite is broken. In the simulation, the PSO algorithm can gain a group of optimal time source combinations within a minimum of 10 iterations, which significantly reduces the computational complexity, shortens the computation time, and can complete the satellite time re-synchronization of the LEO satellite networks more quickly.

Although only 66 satellites are used in the design, the effectiveness of the proposed method has been verified. As the number of satellites increases, the traversal algorithm grows exponentially in computation because it needs to compute all possible scenarios, whereas the PSO algorithm is closer to the optimal result with each iteration, eliminating many useless computations. Therefore, its generalization to giant constellations is also applicable.

5. Conclusions

Interference or space environmental factors may lead to time reference failure or time transfer link fault in the time-varying topology of LEO satellite networks [31]. In order to provide the long-distance time-sensitive services, it is necessary to recalculate the time synchronization transfer paths quickly; the means of a fast time synchronization of large-scale LEO satellite networks based on a bionic algorithm is illustrated in this paper.

The ephemeris information from the navigation message is used to generate the connection table between satellites, which can obtain the current topology of the satellite network according to the connection table calculation results.

In the PSO algorithm, six satellites are selected as the time sources, while the stratum of the time transfer trees are calculated. The minimum sum of the weighted layers is considered as the optimization objective to select the time sources. After the clock layers of the satellites in the network are determined, the time cascades of the satellites can be broadcasted to achieve fast and stable time re-synchronization.

With the Iridium constellation topology, the PSO algorithm is compared with the traditional traversal algorithm. The optimization of PSO with the same result requires only 10 iterations, which is much smaller than the 11^6 iterations of the traversal algorithm, and significantly reduces the computational complexity. In this way, we can realize fast time synchronization in large-scale LEO satellite networks with time-varying topology, reduce clock synchronization levels, and decrease time synchronization jitters.

In previous studies, various methods for the time synchronization of individual satellites are considered. These methods include satellite-to-ground bidirectional time synchronization, satellite-to-satellite time synchronization based on TWRTT technology, and time synchronization using global positioning system (GPS) or BeiDou navigation satellite system (BDS) navigation signals. For large-scale LEO satellite constellations, achieving time synchronization for all satellites is crucial. To ensure effective clock synchronization, the clock source must have high precision, necessitating careful selection of the clock source. Utilizing time synchronization methods based on GPS or BDS has the advantage of minimal error sources and high precision, making them the preferred choices for clock sources.

In this study, we consider scenarios where environmental or human factors may impact the reception of navigation signals. In cases where navigation signals are unavailable or only a subset of satellites can receive them, it becomes necessary to select satellites within the current network as clock sources to accomplish time synchronization. In previous studies, when selecting a satellite as a clock source within a constellation, an iterative algorithm is utilized to compute all possible selection scenarios, compare their computational results, and select the optimal result. This study utilizes an iterative bionic algorithm, which significantly reduces the amount of computation. During simulation, we observed that the PSO algorithm occasionally converges to local optimal solutions without reaching the global

optimum after 100 iterations. In the future, we plan to improve this algorithm or explore alternative algorithms to prevent it from becoming trapped in local optimal solutions.

Author Contributions: Conceptualization, Y.X. and J.Y.; methodology, J.Y.; software, Y.X. and H.J.; validation, Y.X. and J.Y.; formal analysis, T.D.; investigation, T.D.; resources, Z.L.; data curation, Z.Z.; writing—original draft preparation, Y.X.; writing—review and editing, J.Y. and J.W.; visualization, Z.Z.; supervision, T.D.; project administration, T.D.; funding acquisition, T.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by National Key R&D Program of China (2022YFB2902504).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: There is no conflict of interest between any of the authors and the company Space Star Technology Co., Ltd.

Appendix A

In the pseudocode, for, while, if and return are functions defined in C/C++ programming language. Among them, for and while functions choose whether to loop and execute the following program by determining whether the conditions that follow are satisfied or not. The if function is a judgment function. The and return function returns one or more variables and constants. Q is an array to store unvisited satellite numbers. The col function stores other satellite numbers that have a connection relationship with the satellite represented by the first element in the Q array. And the function sum is used to sum the elements of an array. The arrow and the equal sign represent the assignment of the value on the right side of the symbol to the variable on the left side of the symbol.

Algorithm A1: Traversal Algorithm

Input: Connection table, A ; Number of satellites, N ; Number of time sources, D ; Satellites in each orbital plane, SN ;
Output: Sum of weighted clock hierarchy for each combination of time sources, $sumstratum$; Combinations of time sources with the minimum sum of hierarchy, $sour$; Clock transfer trees, $tree$;

```

1 for  $i_1 \leftarrow 1$  to  $SN$  do
2    $Q(1, 1) = i_1$ ;
3   for  $i_2 \leftarrow 1$  to  $SN$  do
4      $Q(1, 2) = i_2 + SN$ ;
5     :
6     for  $i_D \leftarrow 1$  to  $SN$  do
7        $Q(1, D) = i_D + (D - 1) \cdot SN$ ;
8       Mark the time sources as visited;
9       Mark the hierarchy of the time sources as 0;
10      while  $sum(Q)$  do
11         $col = find(A(Q(1), :) = 1)$ ;
12        for  $j \leftarrow 1$  to  $size(col)$  do
13          if  $col(j)$  has not been visited then
14            Mark  $col(j)$  as visited;
15            Update  $q$  by  $col(j)$ ;
16            Increase the hierarchy of  $col(j)$  by 1;
17            Record the parent node of  $col(j)$ ;
18            Create a clock transfer tree path;
19          Calculate  $sumstratum$  by using weighted clock hierarchy for
20          the current combination of time sources;
21  $sour = find(sumstratum = \min(sumstratum))$ ;
22  $tree \leftarrow$  Create the clock transfer trees;
23 return  $sumstratum, sour, tree$ ;
```

In Algorithm A2, S is an array that records the clock hierarchy.

Algorithm A2: Compute the minimum sum of weighted clock levels for a time source combination presented by x

Input: Time source numbers vector, x ; Connection table, A ; Number of satellites, N ; Number of time source numbers, D ;

Output: The sum of clock weighted levels for all satellites, $sumstr$;

```

1 Init  $q = x$ ;
2 Mark the time sources as visited;
3 Mark the hierarchy of the time sources as 0;
4 while  $sum(q)$  do
5    $col = find(A(q(1), :) = 1)$ ;
6   for  $j \leftarrow 1$  to  $size(col)$  do
7     if  $col(j)$  has not been visited then
8       Mark  $col(j)$  as visited;
9       Update  $q$  by  $col(j)$ ;
10      Increase the hierarchy of  $col(j)$  by 1;
11      Record the parent node of  $col(j)$ ;
12      Create a clock transfer tree path;
13 Calculate  $sumstr$  by using weighted clock hierarchy for the current
    combination of time sources;
14 return  $sumstr$ ;
```

Algorithm A3: Main Algorithm

Input: Number of particles, NP ; Connection table, A ; Number of satellites, N ; Number of time source numbers, D ; Inertia weight, ω ; Learning factors, $c1$ & $c2$; Maximum/minimum particle velocity, $Vmax/Vmin$; Maximum/minimum number of every time source, $Xmax/Xmin$; Iteration times, T ;

Output: Global optimal time sources, g ; Global minimum sum of weighted clock hierarchy, $gbest$; Clock transfer trees, $tree$;

```

1  $v \leftarrow$  Init velocity for all individual particles;
2  $x \leftarrow$  Init time source numbers selected by each individual particle;
3  $p \leftarrow$  Init optimal time source for each individual particle;
4  $pbest \leftarrow$  Init minimum sum of weighted clock hierarchy for each
    individual particle using  $func1$ ;
5  $g \leftarrow$  Init global optimal time sources;
6  $gbest \leftarrow$  Init global minimum sum of weighted clock hierarchy;
7 for  $i \leftarrow 1$  to  $T$  do
8   for  $j \leftarrow 1$  to  $NP$  do
9     Calculate  $sumstr = func1(x(j), :, A, N, D)$ ;
10    if  $sumstr < pbest(j)$  then
11      Update calculated optimal time sources  $p(j)$  and minimum
        weighted clock hierarchy for the particle  $pbest(j)$ ;
12    if  $sumstr < gbest$  then
13      Update global minimum weighted clock hierarchy  $gbest$  and
        global optimal time sources  $g(j)$ ;
14    Update the velocity  $v$  and time source satellite numbers  $x$  for the
        particle;
15  $Q = g$ ;
16 while  $sum(Q)$  do
17    $col = find(A(Q(1), :) = 1)$ ;
18   for  $j \leftarrow 1$  to  $size(col)$  do
19     if  $col(j)$  has not been visited then
20       Mark  $col(j)$  as visited;
21       Update  $Q$  by  $col(j)$ ;
22       Increase the hierarchy of  $col(j)$  by 1;
23       Record the parent node of  $col(j)$ ;
24       Create a clock transfer tree path;
25  $tree \leftarrow$  Create the clock transfer trees;
26 return  $g, gbest, tree$ ;
```

References

1. Zhao, Y.; Cao, J.; Li, Y. An Improved Timing Synchronization Method for Eliminating Large Doppler Shift in LEO Satellite System. In Proceedings of the 2018 IEEE 18th International Conference on Communication Technology (ICCT), Chongqing, China, 8–11 October 2018; pp. 762–766.
2. Chefrour, D. Evolution of network time synchronization towards nanoseconds accuracy: A survey. *Comput. Commun.* **2022**, *191*, 26–35. [[CrossRef](#)]
3. Guo, W.; Lu, A.-A.; Gao, X.; Xia, X.-G. Broad Coverage Precoder Design for Synchronization in Satellite Massive MIMO Systems. *IEEE Trans. Commun.* **2021**, *69*, 5531–5545. [[CrossRef](#)]
4. Huang, M.; Chen, J.; Feng, S. Synchronization for OFDM-Based Satellite Communication System. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5693–5702. [[CrossRef](#)]
5. Vaseghi, B.; Hashemi, S.S.; Mobayen, S.; Fekih, A. Finite Time Chaos Synchronization in Time-Delay Channel and Its Application to Satellite Image Encryption in OFDM Communication Systems. *IEEE Access* **2021**, *9*, 21332–21344. [[CrossRef](#)]
6. Marrero, L.M.; Merlano-Duncan, J.C.; Querol, J.; Kumar, S.; Krivochiza, J.; Sharma, S.K.; Chatzinotas, S.; Camps, A.; Ottersten, B. Architectures and Synchronization Techniques for Distributed Satellite Systems: A Survey. *IEEE Access* **2022**, *10*, 45375–45409. [[CrossRef](#)]
7. Zhang, Z.; Wang, D.; Liu, L.; Wang, B.; Sun, C. A Frequency Offset Independent Timing Synchronization Method for 5G Integrated LEO Satellite Communication System. In Proceedings of the 2022 IEEE 22nd International Conference on Communication Technology (ICCT), Nanjing, China, 11–14 November 2022; pp. 423–427.
8. Liu, L.; Zhu, L.-F.; Han, C.-H.; Liu, X.-P.; Li, C. The Model of Radio Two-way Time Comparison between Satellite and Station and Experimental Analysis. *Chin. Astron. Astrophys.* **2009**, *33*, 431–439. [[CrossRef](#)]
9. Koppang, P.; Wheeler, P. Working application of TWSTT for high precision remote synchronization. In Proceedings of the 1998 IEEE International Frequency Control Symposium, Pasadena, CA, USA, 27–29 May 1998; pp. 273–277.
10. Gaglione, S.; Angrisano, A.; Freda, P.; Innac, A. Benefit of GNSS Multiconstellation in Position and Velocity Domain. In Proceedings of the 2015 IEEE Metrology for Aerospace (MetroAeroSpace), Benevento, Italy, 4–5 June 2015; pp. 9–14.
11. Xu, J.; Zhang, C.; Wang, C.; Jin, X. Approach to inter-satellite time synchronization for micro-satellite cluster. *J. Syst. Eng. Electron.* **2018**, *29*, 805–815.
12. Yang, Y.; Gao, W.; Guo, S.; Mao, Y.; Yang, Y. Introduction to BeiDou-3 navigation satellite system. *Navigation* **2019**, *66*, 7–18. [[CrossRef](#)]
13. Wang, J.; Li, L.; Zhou, M. Topological dynamics characterization for LEO satellite networks. *Comput. Netw.* **2007**, *51*, 43–53. [[CrossRef](#)]
14. Bao, J.; Zhao, B.; Yu, W.; Feng, Z.; Wu, C.; Gong, Z. OpenSAN. In Proceedings of the 2014 ACM Conference on SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 347–348.
15. Wang, F.; Jiang, D.; Qi, S.; Qiao, C.; Shi, L. A Dynamic Resource Scheduling Scheme in Edge Computing Satellite Networks. *Mob. Netw. Appl.* **2020**, *26*, 597–608. [[CrossRef](#)]
16. Freris, N.M.; Graham, S.R.; Kumar, P.R. Fundamental Limits on Synchronizing Clocks Over Networks. *IEEE Trans. Autom. Control* **2011**, *56*, 1352–1364. [[CrossRef](#)]
17. Liu, T.; Yu, Z.; Ding, Z.; Nie, W.; Xu, G. Observation of Ionospheric Gravity Waves Introduced by Thunderstorms in Low Latitudes China by GNSS. *Remote Sens.* **2021**, *13*, 4131. [[CrossRef](#)]
18. Stankov, S.M.; Jakowski, N.; Tsybulya, K.; Wilken, V. Monitoring the generation and propagation of ionospheric disturbances and effects on Global Navigation Satellite System positioning. *Radio Sci.* **2006**, *41*, 1–14. [[CrossRef](#)]
19. *IEEE Std 1588b-2022*; IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE: New York, NY, USA, 2022.
20. *ISO/IEC/IEEE 8802-1AS:2021(E)*; Timing and Synchronization for Timesensitive Applications in Bridged Local Area Networks. IEEE: New York, NY, USA, 2021.
21. Liu, C.; Liu, Y. A Real-Time Distributed Algorithm for Satellite Constellation Routing. In Proceedings of the 2018 IEEE 18th International Conference on Communication Technology (ICCT), Chongqing, China, 8–11 October 2018; pp. 9–14.
22. Akram, V.K.; Asci, M.; Dagdeviren, O. Design and Analysis of a Breadth First Search based Connectivity Robustness Estimation Approach in Wireless Sensor Networks. In Proceedings of the 2018 6th International Conference on Control Engineering & Information Technology (CEIT), Istanbul, Turkey, 25–27 October 2018; pp. 1–6.
23. Erciyes, K. *Discrete Mathematics and Graph Theory*; Springer: Cham, Switzerland, 2021.
24. Jang-Ho, S.; Chang-Hwan, I.; Chang-Geun, H.; Jae-Kwang, K.; Hyun-Kyo, J.; Cheol-Gyun, L. Multimodal function optimization based on particle swarm optimization. *IEEE Trans. Magn.* **2006**, *42*, 1095–1098. [[CrossRef](#)]
25. Khan, S.U.; Yang, S.; Wang, L.; Liu, L. A Modified Particle Swarm Optimization Algorithm for Global Optimizations of Inverse Problems. *IEEE Trans. Magn.* **2016**, *52*, 1–4. [[CrossRef](#)]
26. Liu, X.; Zhao, Y.; Zhu, Z.; Liu, H.; Gan, F. Particle Swarm Optimized Compact, Low Loss 3-dB Power Splitter Enabled by Silicon Columns in Silicon-on-Insulator. *Photonics* **2023**, *10*, 419. [[CrossRef](#)]
27. Li, P.; Liu, X.; Chen, H.; Li, B.; Ma, T.; Jiang, W. Optimization of Three-Dimensional Magnetic Field in Vacuum Interrupter Using Particle Swarm Optimization Algorithm. *IEEE Trans. Appl. Supercond.* **2021**, *31*, 1–4. [[CrossRef](#)]

28. Mao, M.; Chang, J.; Sun, J.; Lin, S.; Wang, Z. Research on VMD-Based Adaptive TDLAS Signal Denoising Method. *Photonics* **2023**, *10*, 674. [[CrossRef](#)]
29. Shi, L.; Xie, Z.; Lv, J. Research on LAP routing algorithm based on traversal tree. In Proceedings of the 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, 28–30 May 2016; pp. 2137–2142.
30. Adhitama, M.A.; Sarno, R. Account charting and financial reporting at accounting module on enterprise resource planning using tree traversal algorithm. In Proceedings of the 2016 International Conference on Information & Communication Technology and Systems (ICTS), Tashkent, Uzbekistan, 2–4 November 2016; pp. 20–25.
31. Gao, W.; Li, H.; Zhong, M.; Lu, M. The Separate Clock Drift Matched Filter to Detect Time Synchronization Attacks Toward Global Navigation Satellite Systems. *IEEE Trans. Ind. Electron.* **2023**, *70*, 6305–6315. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.