Computer Systems Science & Engineering
DOI:10.32604/csse.2022.021604
Article

Tech Science Press

N-Body Simulation Inspired by Metaheuristics Optimization

Muhammad Ali Ismail*, **Maria Waqas and Farah Sadiq**

National Center in Big Data and Cloud Computing, Department of Computer and Info Systems Engineering, NED University of Engineering and Technology, Karachi, Pakistan
*Corresponding Author: Muhammad Ali Ismail. Email: maismail@neduet.edu.pk
Received: 08 July 2021; Accepted: 12 August 2021

Abstract: The N-body problem in classical physics, is the calculation of force of gravitational attraction of heavenly bodies towards each other. Solving this problem for many heavenly bodies has always posed a challenge to physicists and mathematicians. Large number of bodies, huge masses, long distances and exponentially increasing number of equations of motion of the bodies have been the major hurdles in solving this problem for large and complex galaxies. Advent of high performance computational machines have mitigated the problem to much extent, but still for large number of bodies it consumes huge amount of resources and days for computation. Conventional algorithms have been able to reduce the computational complexity from $O(n^2)$ to $O(nlogn)$ by splitting the space into a tree or mesh network, researchers are still looking for improvements. In this research work we propose a novel solution to N-body problem inspired by metaheuristics algorithms. The proposed algorithm is simulated for various time periods of selected heavenly bodies and analyzed for speed and accuracy. The results are compared with that of conventional algorithms. The outcomes show about 50% time saving with almost no loss in accuracy. The proposed approach being a metaheuristics optimization technique, attempts to find optimal solution to the problem, searching the entire space in a unique and efficient manner in a very limited amount of time.

Keywords: N-body problem; metaheuristics optimization; particle swarm optimization; heavenly bodies

1 Introduction

1.1 The N-body Problem

The N-body problem is a series of calculations that are performed to find out the gravitational impact of celestial objects on one another and their motion under the influence of this impact. It can be used to find out the position of each object in the universe at any given time, given a few initial conditions. The two-body problem is the most simplified form of the N-body problem where the gravitational impact of only two bodies over one another is observed. Sir Isaac Newton proposed a formula for computing gravitational force between two objects, if the initial position and mass of the objects are known. The formula is given as Eq. (1) [1].

$$F = \frac{G * m_1 * m_2}{r_2^2 - r_1^2} \tag{1}$$

Here G is gravitational constant, its value is 6.67×10^{-11} Nm^2/kg^2, m_1 is the mass of first body, m_2 is the mass of the second body, r_1 is the position of first body and r_2 is the position of second body.

The calculations involved in N-body simulations are computationally very expensive, the reason being exhaustive calculation of interaction of one body with all the other bodies in the system. For n number of bodies, the complexity of computation is $O(n^2)$, which increases exponentially as the number of bodies are increased greatly. This high complexity burns up the computational resources very fast as the number of bodies in the system are increased, hence making it highly unfeasible to carry out simulation of large clusters of celestial objects. This constraint has been proven to be the main hindrance in the way of carrying out large simulation with limited computational resources effectively.

There are two main methods of N-body simulation, direct gravitational N-body simulations [2] and optimization simulation [3]. Direct method, as the name suggests, uses the Newton's equation of gravitational force to compute all the forces within the system whereas optimization simulations use some approximation in their calculations in order to speed up the complex process. The most popular approximation simulations include tree method [4], particle-mesh method [5], fast multi-pole method [6], adaptive mesh method [7] and self-consistent tree method [8].

1.2 Metaheuristics

Metaheuristics is the optimization technique to find approximate solutions to the problems, it does not promise to find the best solution, but it scans the entire search space in a unique and efficient manner that can find good solutions to the problem in a very limited amount of time. Thus it finds out nearly optimal solutions to the problems having a large solution set under constrained computational resources and time, as the solution set of these problems is too huge to be searched using conventional searching methods [9]. The metaheuristics techniques can be applied to our problem because of the similarity in the objective: *Reducing the number of computations to find out the final solution*. The solution may compromise on the accuracy, but it provides near to real results in lesser time and computations.

Some of the most accomplished metaheuristics algorithms include tabu search, simulated annealing, particle swarm optimization and genetic algorithms. Tabu search relies mainly on memory and the search history, which most other contemporary algorithms do not follow [10]. Although mathematical analysis of tabu algorithms could become too complex at times due to the introduction of too many degrees of freedom, still it is one of the most favored metaheuristics optimization technique [11]. Simulated annealing technique is inspired by the metal annealing processing. The biggest advantage it has over gradient based algorithms or deterministic algorithms is that it does not get trapped in local minima [12]. Particle swarm optimization is based on the natural swarm behavior of bird and fish schools [13]. Lastly, genetic algorithms are inspired by Charles Darwin's biological evolution theory of natural selection, crossover, recombination, mutation and selection [14].

This research is an effort to bring together the domains of n-body simulation and metaheuristics and use metaheuristic optimization techniques to efficiently solve the n-body problem. The results effectively show improvement in time and efficiency, thus the technique seem promising in improving the performance of n-body simulation under time and resources constraints. The remaining of the paper is organized as follows: Section 2 covers the literature review in the related areas, Section 3 discusses the developed algorithm and its analogies with existing metaheuristic techniques, Section 4 discusses the results of the simulations, and Sections 5 presents the final concluding remarks. To facilitate readership, all mathematical relations and symbols given in this write up are summarized in Tab. 2, given in the end of the paper.

2 Related Work

The n-body simulation has been a challenge for mathematicians and researchers for many centuries. There have been numerous attempts to develop a fast and efficient algorithm that can effectively carry out simulation of motion of trillions of heavenly bodies present in our universe. One of the methods to carry out simulation is using equation of gravitational force, Eq. (1), to calculate all the interactions between bodies within the system without using any approximation. This method quickly becomes inefficient because of the high complexity, $O(n^2)$, of the equations as the number of bodies increase. Many optimization techniques have also been devised to solve the same problem in reasonable time frame. Mostly, tree algorithms are used. These algorithms split the universe into a tree structure of cells. The particles located at the far away cells are considered to be a single body and the center of mass of the system is used to find the gravitational force. This reduces the complexity of the algorithm to $O(n\ logn)$ [4,15]. Similarly, in mesh method, the universe is divided into a mesh and the potential energy of the surface of the mesh is found out, which is ultimately used to find out the gravitational force [5,16,17]. Another approximation technique is fast-multipole method, in this method it is assumed that particles located close to each other experience same kind of force under the influence of the particles located far away in the universe. This assumption helps in reducing the number of interactions that needed to be calculated to find out the gravitational force of the system [6,18]. Another very recent and popular approach used is parallelization of the n-body simulation. This allows multiple n-body calculations to be carried out simultaneously, hence speeding up the process. For this purpose state-of-the-art hardware infrastructure (like GPUs) is required to carry out parallel processing. Another approach is to exploit distributed or cloud computing in order to achieve parallelism [19,20].

Metaheuristics algorithms have found themselves a large number of applications where they are proving to be very useful in providing optimal solutions to complex problems in limited time and resources. The main reasons of the popularity of these algorithms are the adaptability, ability to produce good approximate results in limited time, ability to find global optima, bringing out of the box solutions and the ease with which they can be applied to any optimization problem. Metaheuristics are being widely used in financial applications like, portfolio selection, portfolio optimization, project scheduling, and predicting bankruptcy [21]. Metaheuristics are also being used in the domain of power systems to optimize the processes like generator maintenance scheduling and optimization of power flow problem [22]. In civil engineering domain, metaheuristic optimization is being used to optimize structure of buildings, optimization of grids, optimization of mechanical domes and optimization of design of floors [23]. It the field of digital signal processing, metaheuristic approach aids in scheduling problems, image processing, very large scale integration design, communication and many other applications [24]. So far we have not been able to find an attempt of solving the n-body problem with the help of metaheuristic optimization. Last but not the least, these algorithms have been able to find way in aiding solutions of NP (non-deterministic polynomial) hard problems in computer system design and networks. Numerous metaheuristic algorithms have been devised to address virtual machine (VM) consolidation problem [25]. Floor planning in VLSI (very large scale integration) and NOC (network-on-chip) layouts is another NP hard problem where researchers have experimented with metaheuristics approaches [26–29].

3 N-Body Simulation Using Metaheuristics

3.1 Inspiration

As stated earlier, the N-body simulation algorithms, inspired by the metaheuristics algorithm of optimization, do not promise to provide the optimal solution but work in an efficient manner to provide a good enough solution in minimal time and resources. These algorithms are inspired by nature and its processes. We take our inspiration from metaheuristics algorithms to give approximate solution of N-body problem using minimum time and computational resources, specifically *particle swarm optimizatiom*.

Particle swarm optimization (PSO) was realized by Kennedy and Eberhart in 1995 [13]. The basic idea of this optimization technique is based on the natural swarm behavior of bird and fish schools. Soon the idea of swarm intelligence gained popularity in the research world, with many applications in the field of optimization, scheduling/design applications and computational intelligence. Currently about two dozens of PSO variants are used. Some hybrids of PSO with other techniques also exist [14]. PSO search keeps track of particles in a given objective function space by adjusting particles' paths. A time-dependent positional vector is used to model piece-wise path traces. The vector representing a swarming particle constitutes two components: one is deterministic and the other is stochastic. Each randomly moving particle i is steered towards a position having the current global best g and the local best x_i which is its own best position. The location of the particle i is updated whenever a better location is found. Hence for every iteration each particle maintains a current best at any time t. The iterations are carried out till a global best is found and the location no longer shows any improvement. Eq. (2) gives the updated velocity vector for an iteration, where x_i and v_i respectively represent the position vector and velocity of particle i.

$$v_i^t = v_i^t + \alpha\epsilon_1\left[g - x_i^t\right] + \beta \in_2\left[x_i - x_i^t\right] \tag{2}$$

The parameters $\propto$ and β are the acceleration constants or learning parameters, which are typically equal to 2 and ε_1 and ε_2 are two random vectors, they take a value between 0 and 1. The initial locations of all the particles must be distributed consistently so that they can cover the whole region. The initial velocity of a particle, can be set to zero as given in Eq. (3). The new position can then be updated by Eq. (4).

$$v_i^0 = 0 \tag{3}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{4}$$

In many variants which extend the standard PSO algorithm, the most noticeable improvement is probably to use an inertia function $\theta(t)$ so that v_i^t replaced by $\theta(t)v_i^t$ where $\theta(t)$ lies between 0 and 1. In the simplest case, the inertia function can be taken as a constant, typically $\theta \in [0.5,0.9]$. As stated earlier the iterations tend to converge the particle movements towards an optimized global value [30].

We attempted to solve N-body problem by picking up the problem solving structure of PSO metaheuristic algorithm and apply it on our problem. We observed that only a few particles have a significant impact on the other particles. This leads to a conclusion that calculating forces applied by insignificant particles is an unnecessary operation as it does not have any impact on the movement of that particle. This gives us the opportunity to apply a metaheuristics-inspired solution, where a few best particles are selected to find out the final results. The choice of such particles is a matter of great concern, as the quality of our solution depends on the quality of our choice, hence only the particles with significant impact on each other particle are considered. An *impact co-efficient* is calculated for each particle. This impact co-efficient is the ratio between the mass of the particle and its distance. Observation and experimentation led us to the conclusion that if the value of the impact co-efficient is two times greater than the mass of the body under consideration, then its impact will be significant enough to be calculated. Thus only *significant* particles for each particle were singled out and their impact on a body was calculated, creating a *mini universe* for each body. This saves us from the tedious work of carrying out calculations on all the bodies for each body, despite its significance.

3.2 Algorithm

The proposed algorithm for N-body simulation is as follows:

1. Initialize a universe of n number of bodies.
2. For each body p:
   a. Initialize a mini-universe for the body p ($mini - universe_p$).
   b. For every body q, calculate the impact co-efficient ($mass_q / \ distance_q^2$) where $p \neq q$.
   c. If the impact co-efficient of the body q is greater than the minimum threshold value, then add the body q to the ($mini - universe_p$).
3. For every body p.
   a. For every body q in $mini - universe_p$, calculate the Gravitational force F_{pq} as given in Eq. (5).

$$F_{pq} = \sum_{q \neq p}^{n} \frac{G * m_p * m_q}{r_q^2 - r_p^2} \tag{5}$$

   b. Sum the forces calculated in step 3(a) to obtain equivalent force F_p on the body as shown in Eq. (6).

$$F_p = \sum_{q \neq p}^{n} F_{pq} \tag{6}$$

   c. Use the force calculated in step 3(b) to calculate the acceleration a_p of the body using Eq. (7).

$$a_p = \frac{F_p}{m_q} \tag{7}$$

   d. Use the acceleration calculated in step 3(b) to calculate the velocity v_p of the body as given in Eq. (8).

$$v_p = v_{p\_old} + \left(\Delta t * a_p\right) \tag{8}$$

   Where Δt is the time step.

   e. Finally, the velocity calculated in step 3(c) is used to calculate the new position d_p of the body as given in Eq. (9).

$$d_p = d_{p\_old} + \left(\Delta t * v_p\right) \tag{9}$$

4. Form a graph of simulation for the positions of all the n bodies in the universe for each time step Δt over the total time t.
5. End.

The pseudo code is given as follows.

Algorithm 1: N-body simulation

```
1: Procedure CALCULATE N-BODY INTERACTIONS
2:   $n \leftarrow number\ of\ bodies$
3:   $m \leftarrow mass\ of\ each\ body$
4:   $v \leftarrow velocity\ of\ each\ body$
5:   $d \leftarrow position\ of\ each\ body$
6:   $a \leftarrow acceleartion\ of\ each\ body$
7:   $\Delta t \leftarrow time\ step$
8:   $\mathrm{G} \leftarrow gravitational - constant(6.67exp - 11)$
9:   for $p = 1 : n$ do
10:    Initialize $mini - universe[p]$
11:    for $q = 1 : n$ do
12:      if $p! = q$ then
13:        $impact.coefficient[q] \leftarrow mass[q]/(distance[q])^{2}$
14:        if $impact.coefficient[q] > mass[p]$ then
15:          $mini - universe[p] \leftarrow \mathrm{q}$
16:  for $p = 1 : n$ do
17:    for $q = 1 : mimi - universe[q]$ do
18:      $F[p][q] \leftarrow sqrt\left(G * mass[p] * mass[q]/(d[p] - d[q])^{2}\right)$
19:      $F(p) + = F[p][q]$
20:      $F(p) + = F[p][q]$
21:      $v + = \Delta t * a$
22:      $d + = \Delta t * v$
23:    Use $d$ and $\Delta t$ for each body to form graphical simulation.
```

3.3 Explanation

The algorithm picks and implements many characteristics of metaheuristics algorithms in order to achieve speed up in calculations. The algorithm first calculates an *impact co-efficient* of each body from every other body. If the impact co-efficient is smaller than the threshold value, the body is ignored and no calculations are done for it; otherwise it is considered a part of a *mini-universe* of that body. The interactions are considered only for the bodies present in the mini-universe of each body. This dramatically reduces the number of calculations, complexity and time required to calculate the particle-particle interactions. This approach simplifies the algorithm and preserve the accuracy of the results to an acceptable level.

3.3.1 The Impact Co-efficient

The impact co-efficient is a value calculated for each body in order to decide whether it is feasible to carry out n body calculations for it. The main idea is to take advantage of the fact that in n body calculations, not all bodies have the same gravitational impact on other bodies, and impact of many bodies can easily be neglected because it is too small to make any impact on the movement of that body. The impact is small because of the two main factors:

1. The mass of the body is too small in comparison with the mass of the body under consideration.
2. The distance of the body is too large from the body under consideration.

The impact co-efficient α is heavily dependent on these two factors. As is obvious from the above discussion, Eqs. (10) and (11) can be deduced.

$$\alpha \propto m \tag{10}$$

$$\alpha \propto 1/d^2 \tag{11}$$

The above equations when combined, give us the value of our impact co-efficient α as Eq. (12).

$$\alpha = m/d^2 \tag{12}$$

3.3.2 Calculations of the Impact Co-efficient

The impact co-efficient is calculated for each body in each iteration. It is then compared with a threshold value. The threshold is different for each body and it determines the level of relaxation we are ready to give in the calculations in terms of accuracy. Higher the value of threshold, lower the accuracy and lower the execution time. Lower the value of threshold, higher the accuracy and consequently, higher the execution time.

As the value of threshold is decreased, the system tends to mimic the conventional algorithm. On the other hand, as the value of the threshold is increased, computations become less and less accurate. This parameter is dependent on application requirements for which the algorithm is being used. The more the fluctuation on accuracy can be tolerated, the higher the value of the threshold can be set. But if the system is very sensitive, then a smaller value of the threshold must be used but at the cost of increased computation time and effort.

3.4 Analogy with Metaheuristics Algorithm

The algorithm is inspired by the metaheuristics optimization algorithms. It has taken many techniques used in these algorithms to make the code efficient and fast. As in genetic algorithms of metaheuristics optimization, a population of individuals is first chosen from a set of individuals, the same way, the number of bodies are chosen for each body. The selection is done based on some criteria. In our N-body algorithm, the criteria is to compare each body's impact co-efficient with a threshold value. If the value is greater than or equal to the threshold, the body is selected to be the part of calculations for that body.

4 Results and Discussion

The proposed approach has been validated in two aspects. Firstly, part of solar system has been simulated to determine reliability of the algorithm. Secondly, the algorithm is run to find gravitational force between different numbers of randomly generated bodies, and the results are compared with that of conventional algorithm to validate the efficiency of the proposed algorithm. The configuration of the system used is Intel core i7 3.20 GHz processor and 16 GB RAM.

In order to test reliability of the proposed algorithm, trajectories of four selected planets around the sun in the solar system have been plotted and the execution time is measured for four cases: 365 days or 1 year, 273.75 days or 0.75 year, 182.5 days or 0.5 year, and 91.25 days or 0.25 year. Time step is kept same for all the cases and is taken to be 7 hours or 25000 seconds. The selected planets are Mars, Earth, Venus and Mercury. Fig. 1 shows the resulting plots. The outer most trajectory is that of Mars, then Earth, then Venus and then Mercury. The inner most spot is the sun. For 1 year, the calculation took 0.9006 seconds. The results are shown in Fig. 1(a). For 0.75 year, the calculation took 0.7489995956420898 seconds to complete. Results are shown in Fig. 1(b). It can be seen in the figure, Earth has completed 75% of its orbit showing that simulation is satisfactorily reliable. For 0.5 year, the calculation took 0.448799 seconds to complete. Results are shown in Fig. 2(c), showing that the Earth has completed 50% of its orbit. For 0.25 year, the calculation took 0.22340 seconds to complete. Results are shown in

Fig. 2(d), again showing that the Earth has completed 25% of its orbit. These results show that the proposed algorithm is able to simulate the selected portion of the solar system with satisfactory accuracy.

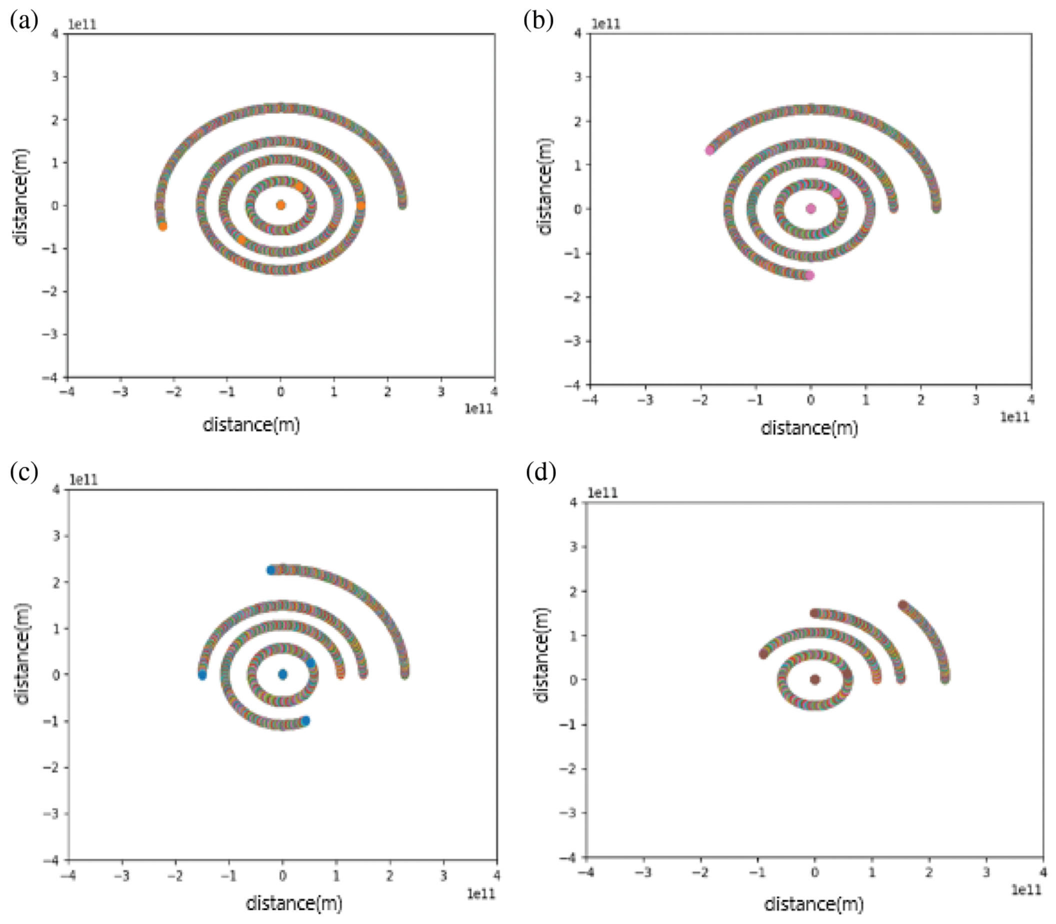

Figure 1: Trajectories for different Earth years (a) 1 Earth year (b) 0.75 Earth year (c) 0.5 Earth year (d) 0.25 Earth year

After testing the reliability of the algorithm, the same was tested for time efficiency over the conventional approach. For this purpose, both the proposed metaheuristic algorithm and the conventional algorithm were run to find gravitational force for different time spans for randomly generated bodies. The conventional algorithm uses Eq. (1) to calculate all the gravitational interactions of each body with every other body in the system. Unit masses were taken for the bodies and the value of *G* was set to 1 for both the algorithms. The process was run ten times for different numbers of bodies, which are 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. Each set is simulated with time step of 7 hours for the duration of 1,2,3,4 and 5 years.

The results have been tabulated in Tab. 1. The table also shows speedup of the proposed metaheuristic algorithm over the conventional one. It can be seen that the speedup is over 50% for every run, and it keeps on improving with increasing number of bodies. As can be seen the speed up calculated for 100 bodies is 57.36% on average. The speed up obtained for 200 bodies is found to be 55.7% on average. The speed up obtained for 300 bodies is found to be 57.99% on average. The speed up obtained for 400 bodies is found to be 75.22% on average. The speed up obtained for 500 bodies is found to be 76.76% on average.

The speed up obtained for 600 bodies is found to be 84.48% on average. The speed up obtained for 700 bodies is found to be 84.41% on average. The speed up obtained for 800 bodies is found to be 83.77% on average. The speed up obtained for 900 bodies is found to be 83.73% on average. The speed up obtained for 1000 bodies is found to be 82.47% on average. Thus the trends shown in Tab. 1 suggest that the proposed algorithm is roughly two times faster than the conventional algorithm.

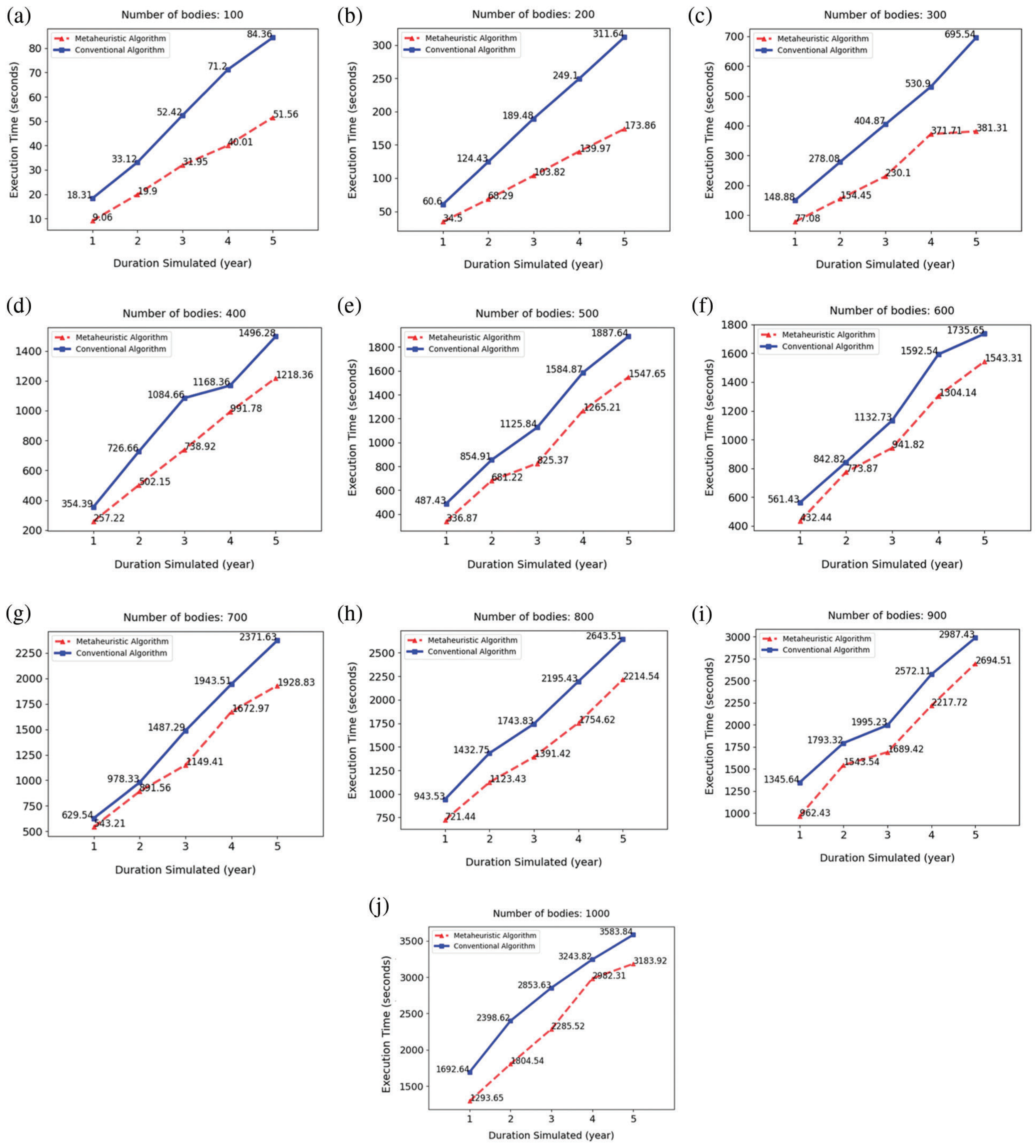

Figure 2: Comparison of execution times for conventional and metaheuristics algorithm for different number of bodies (a) 100 bodies (b) 200 bodies (c) 300 bodies (d) 400 bodies (e) 500 bodies (f) 600 bodies (g) 700 bodies (h) 800 bodies (i) 900 bodies (j) 1000 bodies

Table 1: Speed comparison results between execution times for proposed metaheuristic algorithm (MH) and conventional algorithm (C) for different durations

No of Bodies	Execution Time (seconds)										Speed up
	1 year		2 year		3 year		4 year		5 year		
	MH	C	MH	C	MH	C	MH	C	MH	C	
100	9.06	18.31	19.9	33.12	31.95	52.42	40.01	71.2	51.56	84.36	57.36%
200	34.5	60.6	68.29	124.43	103.82	189.48	139.97	249.1	173.86	311.64	55.70%
300	77.08	148.88	154.45	278.08	230.1	404.87	371.71	530.9	381.31	695.54	57.99%
400	257.22	354.39	502.15	726.66	738.92	1084.66	991.78	1168.36	1218.36	1496.28	75.22%
500	336.87	487.43	681.22	854.91	825.37	1125.84	1265.21	1584.87	1547.65	1887.64	76.76%
600	432.44	561.43	773.87	842.82	941.82	1132.73	1304.14	1592.54	1543.31	1735.65	84.48%
700	543.21	629.54	891.56	978.33	1149.41	1487.29	1672.97	1943.51	1928.83	2371.63	84.41%
800	721.44	943.53	1123.43	1432.75	1391.42	1743.83	1754.62	2195.43	2214.54	2643.51	83.77%
900	962.43	1345.64	1543.54	1793.32	1689.42	1995.23	2217.72	2572.11	2694.51	2987.43	83.73%
1000	1293.65	1692.64	1804.54	2398.62	2285.52	2853.63	2982.31	3243.82	3183.92	3583.84	82.47%

Table 2: Summary of mathematical relations and notations

Notation	Explanation
F	Gravitational force between two objects calculated as $\frac{G * m_1 * m_2}{r_2^2 - r_1^2}$.
G	Gravitational constant with value $6.67 \times 10^{-11} Nm^2/kg$
m_1, m_2	Masses of the two bodies
r_1, r_2	Position of the two bodies
i	A random particle in the swarm
g	Global best position of a particle
x_i	Local best position of the particle i
v_i	Velocity of particle i
x_i^t	Local best position of particle i at time instance t
v_i^t	Velocity of particle i at time instance t
α, β	Acceleration constants
$\in_1, \in_2$	Random vectors
$\theta(t)$	Inertial function
α	Impact coefficient of a body with mass m and distance d, calculated as $\frac{m}{d^2}$
F_{pq}	Gravitational force between particles p and q calculated as $\sum_{q \neq p}^{n} \frac{G * m_p * m_q}{r_q^2 - r_p^2}$
m_p, m_q	Masses of the two bodies p and q respectively
r_p, r_q	Position of the two bodies p and q respectively
F_p	Equivalent for on particle q calculated as $\sum_{q \neq p}^{n} F_{pq}$
a_p	Acceleration of body p calculated as $\frac{F_p}{m_q}$
Δt	Time step
v_p	Velocity of body p calculated as $v_{p\_old} + (\Delta t * a_p)$
d_p	Position of body p calculated as $d_{p\_old} + (\Delta t * v_p)$

Same results have been presented graphically in Fig. 2. Ten plots have been generated for different number of bodies (100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000) as shown in the figure. The x-axis of each plot represents the time span or the duration in years for which the simulation was run, while the y-axis shows the corresponding execution time in seconds. The solid blue line shows the resulting line graph for the conventional algorithm, and the dashed red line shows the same for the proposed metaheuristic algorithm. As can be seen in all the plots that the line graph of metaheuristic algorithm never crosses and in most cases is well below the line graph of conventional algorithm. It can also be observed that in many of the sub-plot the gap between the two line graphs widens with increasing number of years (simulated duration), signaling performance improvement with passage of time. Hence from these visual representations as well the proposed approach seems better and promising over the conventional one.

Thus the above two-fold analysis brings us to the conclusion that the metaheuristics-inspired simulations take roughly half the time as compared to the conventional algorithm with reasonable accuracy and reliability.

5 Conclusion

The objective of this study was to devise a novel methodology to carry-out N-body simulations effectively in limited time and with limited resources. An attempt has been made to design such an algorithm using metaheuristics technique based on particle swarm optimization. The proposed algorithm has been validated both for reliability and performance and is found to be more efficient than the conventional approach. The proposed technique can be further improved in two ways. One of the main features that can be added to improve efficiency while preserving the speed up is to define rules to explicitly include some particles in the mini universe irrespective of their impact co-efficient. This approach will let us improve the accuracy by including more particles in the mini-universe. It will not have a huge impact on execution time, because only a few additional particles will be considered. The efficiency of the algorithm can be further improved to great extent by parallelization of the code. Depending on situation, it can lead to decrease computation complexity of upto less than $O(n)$, since for each particle, number of computations are less than n, this approach can result into very small execution times.

Acknowledgement: This research study has been supported by National Center in Big Data and Cloud Computing, NED University of Engineering and Technology, Karachi, Pakistan.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] I. Newton, *The mathematical principles of natural philosophy: Philosophiæ naturalis principia mathematica*. Scotts Valley, California, United States: CreateSpace Independent Publishing Platform, 2017.

[2] S. Mikkola and S. J. Aarseth, "An implementation of N-body chain regularization," *Celestial Mechanics & Dynamical Astronomy*, vol. 57, no. 3, pp. 439–459, 1993.

[3] M. Trenti and P. Hut, "N-body simulations (gravitational)," *Scholarpedia*, vol. 3, no. 5, pp. 3930, 2008.

[4] J. Barnes and P. Hut, "A hierarchical O (N log N) force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986.

[5] P. Bodenheimer, G. P. Laughlin, M. Rozyczka, T. Plewa, H. W. Yorke *et al., Numerical Methods in Astrophysics: An Introduction*. 1st ed., Boca Raton, Florida, United States: CRC Press, 2006.

[6] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 135, no. 2, pp. 280–292, 1997.

[7] G. L. Bryan and M. L. Norman, "Statistical properties of x-ray clusters: Analytic and numerical comparisons," *Astrophysical Journal*, vol. 495, no. 1, pp. 80–99, 1998.

[8] L. Hernquist and J. E. Barnes, "Are some N-body algorithms intrinsically less collisional than others?," *Astrophysical Journal*, vol. 349, pp. 562–569, 1990.

[9] M. Abdel-Basset, L. Abdel-Fatah and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, Cambridge, Massachusetts, United states: Academic Press, pp. 185–231, 2018.

[10] F. Glover, "Tabu search—part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[11] D. Gamboa, C. Rego and F. Glover, "Data structures and ejection chains for solving large-scale traveling salesman problems," *European Journal of Operational Research*, vol. 160, no. 1, pp. 154–171, 2005.

[12] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. of ICNN'95-Int. Conf. on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[14] Y. Jiang, T. Hu, C. Huang and X. Wu, "An improved particle swarm optimization algorithm," *Applied Mathematics and Computation*, vol. 193, no. 1, pp. 231–239, 2007.

[15] J. S. Bagla, "Cosmological N-body simulation: Techniques, scope and status," *Current Science*, vol. 88, no. 7, pp. 1088–1100, 2005.

[16] P. E. Kyziropoulos, C. K. Filelis-Papadopoulos and G. A. Gravvanis, "N-body simulation based on the Particle Mesh method using multigrid schemes," in *2013 Federated Conference on Computer Science and Information Systems*, Krakow, Poland, pp. 471–478, 2013.

[17] L. H. Garrison, D. J. Eisenstein, D. Ferrer, J. L. Tinker, P. A. Pinto *et al.,* "The abacus cosmos: A suite of cosmological *N* -body simulations," *Astrophysical Journal Supplement Series*, vol. 236, no. 2, pp. 43, 2018.

[18] Q. Wang, "A hybrid Fast Multipole Method for cosmological N-body simulations," *Research in Astronomy and Astrophysics*, vol. 21, no. 1, pp. 003, 2021.

[19] P. Liu and S. N. Bhatt, "Experiences with parallel N-body simulation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 12, pp. 1306–1323, 2000.

[20] P. Fortin and M. Touche, "Dual tree traversal on integrated GPUs for astrophysical *N*-body simulations," *Int. Journal of High Performance Computing Applications*, vol. 33, no. 5, pp. 960–972, 2019.

[21] A. Soler-Dominguez, A. A. Juan and R. Kizys, "A survey on financial applications of metaheuristics," *ACM Computing Surveys (CSUR)*, vol. 50, no. 1, pp. 1–23, 2017.

[22] E. Cuevas, E. B. Espejo and A. C. Enríquez, *Metaheuristics algorithms in power systems*, 1st ed., United States: Springer International Publishing, 2019.

[23] A. Kaveh, *Applications of metaheuristic optimization algorithms in civil engineering*, 1st ed., United States: Springer International Publishing, 2016.

[24] A. Gogna and A. Tayal, "Metaheuristics: Review and application," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 503–526, 2013.

[25] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," *Cluster Computing*, vol. 24, pp. 1–24, 2021.

[26] M. Darbandi, A. R. Ramtin and O. K. Sharafi, "Tasks mapping in the network on a chip using an improved optimization algorithm," *Int. Journal of Pervasive Computing and Communications*, vol. 16, no. 2, pp. 165–182, 2020.

[27] L. L. Laudis, N. Ramadass, S. Shyam, R. Benschwartz and V. Suresh, "An adaptive symbiosis based metaheuristics for combinatorial optimization in VLSI," *Procedia Computer Science*, vol. 167, no. 8, pp. 205–212, 2020.

[28] S. N. Hussain and K. H. Kishore, "Heuristic approach to evaluate the performance of optimization algorithms in VLSI floor planning for ASIC design," *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*, vol. 885, pp. 213–225, 2020.

[29] M. Kaur and P. K. Sharma, "On solving partition driven standard cell placement problem using firefly-based metaheuristic approach," *Int. Journal of Bio-Inspired Computation*, vol. 9, no. 2, pp. 121–127, 2017.

[30] X. S. Yang, "Nature-inspired optimization algorithms: Challenges fand open problems," *Journal of Computational Science*, vol. 46, pp. 101–104, 2020.