Physics-inspired Models for Agile Code and Data in Federated Edges

Bongjun Ko*, Brent Kraczek‡, Theodoros Salonidis*, Prithwish Basu†,
Kevin S. Chan‡, Thomas La Porta§, Andreas Martens¶, James Lambert‖

*IBM T. J. Watson Research, ‡US Army Research Laboratory, †Raytheon BBN Technologies
§Penn State University, ¶IBM United Kingdom, ‖Defence Science and Technology Laboratory

***Abstract*—We study the problem of flexibly, dynamically, and adaptively moving, positioning, and instantiating computing tasks and data in federated, distributed edge systems. We call this process "agile code and agile data" (ACAD). We explore the adaptation of physics-inspired models, used for atomistic simulations, to the ACAD problem, treating the code and data as particles on a graph, interacting through different potential energy models. We discuss the mapping between the different elements of ACAD problem and our particles-on-a-graph model, considering different frameworks for data analytics. We explore gravitational, elastic and Coulombic models, both with global and local energy minimization, finding that the Coulombic model obtains the most efficient solution.**

***Keywords*-agile code; physical models;**

I. Introduction

The proliferation of smart devices makes it increasingly possible to exploit their computing, networking, and storage capacities for executing various computing tasks at scale at the network edge [1]. For example, many data analytics tasks that are traditionally performed on large centralized computing infrastructures can be distributed over these edge devices, so that the analytics can be performed in locations where vast majority of the data is generated. This can improve latency, data security, as well as the scalability in their execution. This approach can also be used to deal with limitations on processing and storage resources. However, these scenarios typically consist of networks with limited bandwidth. We envision such "smart edge" systems can and will be adopted, not only for a single, isolated domain, but also in federated environments in which edge system resources can be leveraged across the domain boundaries for running the computing tasks on the data at the edge.

We study the problem of flexibly, dynamically, and adaptively moving, positioning, and instantiating computing tasks and data in federated, distributed edge systems. Such mechanisms allow efficient and flexible utilization of limited system resources of individual edge devices. They also facilitate dynamically handling the fluctuation and dynamics of the available resources and workloads. Finally, they provide a larger degree of freedom in determining where the tasks and data are to be positioned under the data and resource governance policies across the federation boundaries. We use the term "agile code and agile data" (ACAD) to refer to the elements and systems in this scenario.

In this paper, we identify physically-inspired models (akin to [2]) that emulate the interactions of the ACAD elements in distributed edge systems, to help understand and further control the dynamics of their movement and placement in relation to each other. Physics provides a rich set of mathematical models to explain the various types of interactions–forces or related potential energies–between particles of various types such as masses and charges. Inspired by the empirical potentials developed to capture the essential physics of atomistic systems [3], [4], [5], we seek empirical models to characterize the fundamental interactions between ACAD elements in smart edge computing systems. Specific laws of physics typically govern the movements and locations of individual objects within an interactive physical system. By employing similar models in distributed edge computing systems, we can identify effective, decentralized mechanisms that determine the movements and placement of ACAD elements, driven naturally by the "physical" forces they are subjected to.

We first present a taxonomy of the essential elements in distributed edge computing systems (computing tasks, data objects, and system resources) and their properties. We then identify different types of the interactions between them and develop the notion of an interaction potential, or simply "potential", which is a measure of the full set of interactions needed to model the system. We examine a few different types of well known potential energy models, e.g., gravity, electro-static potential, elastic forces, etc. [5]. We then present a simple framework of applying the potential, employing potential energy minimization to equilibrate a test system of agile tasks and data, in which each object in the system moves to minimize interaction energy between it and other objects in the system, as determined by the potential. This is similar to energy minimization in molecular simulations, corresponding to a zero-temperature solid. We also present a sufficient condition in which the decentralized movements of the objects will converge to an equilibrium state where no objects would move further. While all the physical models implemented can be used to represent some form of attractive or repulsive force (or their combination) between ACAD objects, we find that different models lead to different equilibrium configurations in the network and

different convergence properties.

With the insights obtained about properties of different physical models, we then provide a use case of applying the electro-static model to distributed data analytics applications. The objective there is to assign appropriate amounts of electric "charges" to the computing and data objects as well as the computing resources, such as CPU and GPU, and data resources, such as storage and memory, in such a way that the interactive properties (object-to-object, object-to-resource) can be satisfied according to their relative attractive or repulsive forces (Figure 1). Through a simulation study, we show that the decentralized movements of the objects due to the potential minimization framework, applied on a graph of the resources, result in desired equilibrium properties such as balanced resource utilization as well as improved affinity (in distance) between the computing tasks and data. We also evaluate the convergence properties (e.g., number of iterations, distance of object movements) of the decentralized process under different types of movement strategies.

II. Computation model and Use Cases

In this section, we first present a taxonomy of computation model that captures relationships of computing tasks, data, and resources in ACAD systems. This computation model of the interaction between different components is represented by physical models in potential functions. We then present as use cases a few examples of existing data processing frameworks and applications that fit this model.

A. Computation model

An application is represented by a set of computing tasks, or simply referred to as 'codes', composing it. Codes are considered atomic operations and may have one or more data inputs and outputs. For each application there exists a set of data that these codes operate on. During application execution, instances of these codes and data are deployed, moving and executed in a network of physical nodes, which represent computing and data resources such as available CPU, memory and disk as well as network characteristics such as bandwidth and delay between them.

The interaction between the individual components in ACAD systems can be represented by *attractive* or *repulsive* forces, depending on the requirements and operating conditions of the application, where the strength of the forces represent various characteristics about the relationship, e.g., affinity, correlation, (in)compatibility, etc. We distinguish the following types of relationships between codes, data, and physical resources.

1) **Code-Code (C-C) relationship**: An attractive C-C typically indicates a strong dependency between the two tasks, such as one task consuming as input a large size of output from the other. On the other hand, a repulsive C-C can represent the cases of two identical, replicated tasks to be placed apart from each other.
2) **Code-Data (C-D) relationship**: An attractive C-D signifies the physical affinity such as amount of data and bandwidth required by a computing task from a data object, the semantic affinity reflecting the importance of data to the task, or the quantitative affinity such as the cost of executing the task on the data. The C-D relationship can also be effectively used to govern the policies in coalition scenarios, regarding which pieces of data are accessible by which computing tasks; for example, a repulsive force can be put in place between a computing task and a data object if they belong to two different domains and would impose security risks if given access to each other.
3) **Data-Data (D-D) relationship**: If two data objects are frequently processed together, they would have a strong attractive force between them, so they can be placed close to each other. On the other hand, two identical pieces of data (i.e., replicas) would have a very high degree of correlation and for load balancing purposes they should be placed far apart from each other via repulsive force.
4) **Code-Resource (C-R) and Data-Resource (D-R) relationships**. An attractive force between code/data objects and a physical resource would exist if the tasks can be executed, or data can be stored at that physical resource. This relationship can also represent security constraints, compute/storage resource matching constraints, and semantic relationship reflecting a physical node's preference for executing particular types of codes, or storing data, for which various combination of attractive and repulsive forces can be effectively used.

B. Example use cases

Our model can capture a wide range of existing computation frameworks and analytic applications, which would benefit from moving, instantiating, and executing the tasks and data in agile manner. Some examples are listed below.

- In big data frameworks (e.g., MapReduce, Spark), the data is typically large and is impractical to move from the physical nodes they are stored in. Instead, computing tasks are moved to the physical nodes where computation will take place.
- Data stream processing frameworks (e.g., Apache STORM, IBM InfoSphere Streams, Spark) move data instead of computations during application execution. More specifically, analytics are deployed on physical nodes and data in the form of streams are routed to these physical nodes for processing.
- Distributed machine learning applications involve data and analytics that are distributed on multiple physical nodes. Since the model training is typically performed

using iterative methods such as gradient descent, for training machine learning applications, our model can be employed by a physical system where data objects correspond to training data and computing tasks correspond to the training tasks.

- Distributed classification applications, such as face recognition, voice recognition, and object recognition, involve classifying input data (e.g., face images) based on a trained model (e.g. face recognition model) or reference data (e.g., face database). The application graph for such applications typically involves a source data vertex, a feature extraction analytic vertex and a matching analytic vertex which may be connected to a data vertex, representing the reference data.
- Distributed analytics in tactical networks is a proposed future capability in which the dismounted soldiers will carry sensing, computing and processing capabilities that can be harnessed for execution of advanced analytics. Soldiers will collaborate and perform even more complex distributed analytics functions. This is in contrast to current capabilities where soldiers have little individual computing capability and must rely on the Forward Operating Base or other assets to perform such operations. Such capabilities have the potential to significantly improve agility in tactical operations by harnessing the aggregate computing and storage resources available.

III. Physiccal models for ACAD Systems

In this section, we introduce a potential energy model which models the ACAD elements in our distributed computing system as particles interacting via potential energy functions. We then describe a distributed potential minimization framework, each particle moves toward the direction of minimizing its potential energy due to other particles. Note that this corresponds a zero-temperature system in physics terms, while the application of commonly-used methods for finite-temperature atomistic simulations, such as molecular dynamics, Monte Carlo or simulated annealing [6], [3], [4], [5] is left for future work.

A. Potential Energy Model

We model the codes, data, and resources as particles interacting through an interaction potential. This model is inspired by atomistic models used in molecular physics. There are, however, two main differences between atomistic models and the ACAD models presented here. First, particles in atomistic models are moving in a continuous coordinate space, whereas ACAD particles are restricted to discrete locations of the physical nodes that host them. Second, atoms cannot occupy the same position in space, while we allow multiple ACAD particles to occupy the same location on the graph.

In order to account for these differences, we develop a discretized potential energy model, in which the physical metaphors (energy, forces, distance measures, etc.) are defined on graphs of the resources. We assume that the particles occupy discrete positions on a graph $G = (V, E)$, where V is the set of nodes that determine possible locations of the particles and E is the set of edges that represent affinities. Let $M = 1, 2, \cdots, |M|$ denote the set of all ACAD particles in the system. The pair-wise interaction at discrete time t between two particles i and j is modeled by a potential function $u_{i,j}(t)$, which corresponds to a type of force field and is generally a function of the distance $d_{ij}(t)$ between i and j.

The *aggregate* potential energy experienced by a particle j at time t is given by $U_j(t) = \sum_{i \neq j} u_{i,j}(t)$. Since the positions of the particles are affected by their potential energy, the resulting movement of individual particles in turn changes the potential energy of the full system, $\Phi(t) = \frac{1}{2} \sum_j \sum_{i \neq j} u_{i,j}(t)$.

Let $P : M \rightarrow V$ be a mapping of each particle in M onto a node in V, that is $P(i) = v$ means a particle i is at node v in graph G. Given a placement P_t, denoting the placement at time t, the potential energy $u_{i,j}(t)$ can be expressed by $U_j(P_t(i), P_t(j))$ to account for its dependency to the particular placement of the two particles at time t. Similarly, we can re-write $\Phi(t)$, the aggregate potential energy of the system, by Φ^{P_t} to express its dependency to the particular placement of all particles in the system. In general, we write by Φ^P the aggregate potential energy of a system due to a placement P.

B. Distributed Potential Minimization

Given a physical model defined by particles M, node locations G, and potential functions $u_{i,j}(t)$ between particles, one objective is to find distributed movement strategies for particles that result in their optimal placement on node locations in G.

Definition 1: A placement P^* of particles is *optimal* if there is no other placement P such that $\Phi^P < \Phi^{P^*}$.

Definition 2: A placement P_t at time t is said to be an *equilibrium* if the deviation of placement of any particle from P_t at time $t + 1$ can only result in increase in the aggregate potential energy.

Note that the optimal placement is always an equilibrium, but not vice versa; a particle system can reach an equilibrium that is only a local optimum.

We now describe a distributed potential minimization framework which aims at yielding an optimal placement. This is a greedy hill climbing strategy where, at each time step one particle "moves" toward the direction of minimizing its local potential energy resulting from the force fields of other particles. According to this framework, one

particle "moves" at each time step toward the direction of minimizing its local potential energy resulting from the force fields of the other particles. In order to limit the search space for future potential energy calculations, at each time step t, we allow only *one* particle to move to the location of the graph G that minimizes its potential energy, while the locations of all other particles remain fixed. More formally, a particle j moves from a node v at time t to another node v' at time $t+1$ if $v' = \text{argmin}_{w \in N_v} U_j^w(t+1)$, where N_v is a set of candidate nodes w.r.t j's current v at t, and $U_j^w(t+1)$ is the potential energy of j when $P_{t+1}(j) = w$ and $P_{t+1}(i) = P_t(i)$ for all other particles i.

The following convergence property holds with the above strategies for the particle movements:

Proposition 1: Given any initial placement P_0, the system of particles converges to an equilibrium in a finite number steps if the potential energy function between any pair of particles is symmetric, i.e., $u_{i,j}(t) = u_{j,i}(t)$ for all pairs of particles (i, j).

Proof: Only a sketch is provided: The symmetry of the pair-potential energy and the movement of one particle at a time ensures that the aggregate potential energy of the entire system decreases monotonically every time a particle changes its position to some other node that will decrease its individual potential energy. Since there is only a finite number of the possible locations for the particles to be placed, and hence only a finite number of different values of aggregate potential energy can exists, the system will reach a placement that the aggregate potential energy cannot decrease any more, i.e., an equilibrium. ■

C. Assigning Force Fields for ACAD

While we acknowledge that the force-field model for ACAD does not necessarily have to exactly obey those of physical worlds, our starting point is to use some of widely-used models in physics, with the goal of gaining insights on how different types of force models affect the movement and placement of the codes and the data in distributed networks. We evaluate the following three force-field models:

- Gravitational field: $u_{i,j}(t) = \frac{G m_i m_j}{d_{i,j}(t)}$, where m_i and m_j are the measure of masses of particles i and j, respectively, and G is a gravitational constant.
- Electro-static field: $u_{i,j}(t) = -\frac{k_e q_i q_j}{d_{i,j}(t)}$, where q_i and q_j are the measure of electric charges of i and j, respectively, and k_e is a proportionality constant (known as the Coulomb's law constant).
- Elastic field: $u_{i,j}(t) = \frac{1}{2} k_{ij} d_{i,j}(t)^2$, where k_{ij} is a constant w.r.t. the two particles i and j, representing the strength of the elasticity between them.

A critical question that arises is how to assign the parameters of each model to entities represented by the particles, i.e., what value of mass m_i or charge q_i shall be given to each particle i and how the distance between two particles shall be determined, with respect to the properties of the relationship between different entities of ACAD as described in Section II-A. While this is an avenue of our future deeper exploration, here we consider an assignment that will satisfy the following basic principles:

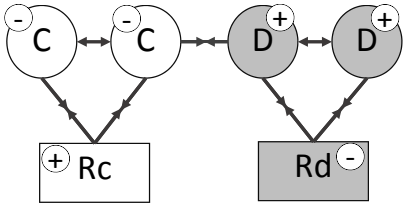


Figure 1. Assignment of electro-static charges.

- A C-D relationship is generally attractive, i.e., analytics and data would be drawn to each other.
- C-R and D-R relationships are attractive, that is, analytics are drawn to computing resources that can execute them, and data are drawn to data storage resources that can host them. To distinguish these two different relationships, we further divide the resources into two types: R_c for computing resources and R_d for data storage resources.
- C-C and D-D relationships are generally repulsive, that is, it would be desirable to produce a load-balancing placement of the same types of analytics (or data) in the system by exerting repulsive force between them.

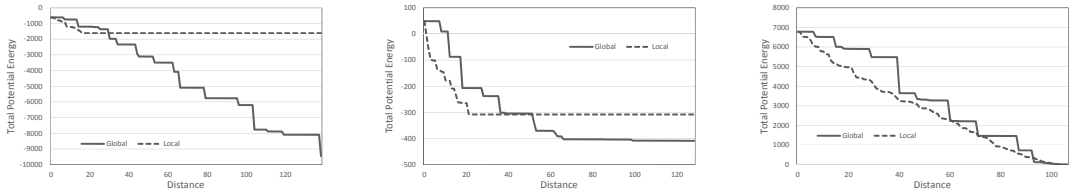
The distance $d_{i,j}$ should reflect both physical distance and semantic distance. In this paper, we make a simplifying assumption that both distance types are already embedded in the given structure of the graph $G = (V, E)$, and hence we simply use the distance in G between any two particles; that is, $d_{i,j}(t) = d_G(P_t(i), P_t(j))$, where $d_G(v, w)$ denotes a graph distance of choice (e.g., shortest path distance) between two nodes v and w in graph G.[1]

Figure 1 illustrates an example of how the electro-static force model can be used to satisfy the above principles by assigning appropriate amount of positive and negative electric charges to different types of entities in ACAD. Such assignment uses the property that, in electro-static model, particles of the same polarity would push each other away, and those of opposite polarity would pull each other.

IV. Experimental Results

In this section, we present preliminary experimental results of particle movements for ACAD system. The experiments are performed by letting the particles representing the codes and data move according to the potential minimization framework in 1-dimensional graphs with unit edge weights. For the distance $d_{i,j}$, we use the shortest-path distance in

[1]One slight modification in the distance measure, deviating from either physical distance or graph distance, is that the distance shall not be 0 in some of the force-field models defined above in order to avoid singularities.



(a) Gravitational Energy (b) Electro-static Energy (c) Elastic Energy

Figure 2. Total System Potential Energy Convergence

the graph, that is the number of hops between the nodes that two particles are located at, with a small constant $\epsilon = 0.01$ is added when they are at the same node to avoid the singularity of the gravitational and electro-static force fields.

In terms of how each particle evaluates the candidate nodes N_v to move to at each step of the movement, we compare the following two methods, differing by how "far" the particles are allowed to move to at each step in the underlying graph:

- **Local search**: each particle is allowed to move only to one of the neighbor nodes to its current location.
- **Global search**: each particle can move (or 'jump') to any node in the graph at each step.

In essence, the local search method is similar in spirit to how molecular dynamics (MD) approaches simulate each particle's movement, and has the advantage over the global one as it requires each particle only to evaluate immediate neighbor nodes as a potential next destination. However, its disadvantage is that the particles can fall into local minima in the potential field (a.k.a. potential "wells") more easily than the global search method, which can often jump over potential "hills" to remote nodes.

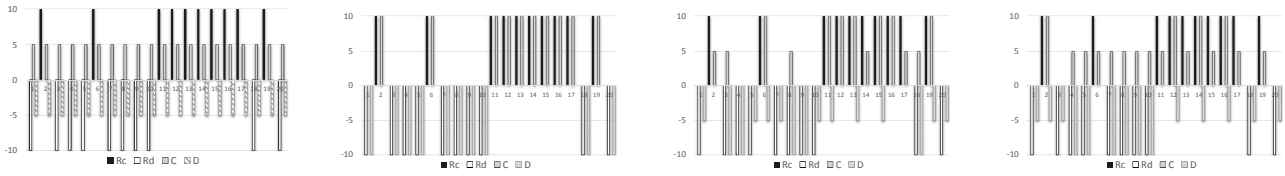
In Figure 2, we show the evolution of the potential energy of the whole system, $\Phi(t)$, as the particles move their locations due to the above two movement methods in a sample run of the simulation; x-axis represents the cumulative distance (i.e., the number of hops on the graph) that the particles move on the graph. Here we simulate a 1-D graph of 20 nodes with 20 particles initially placed at random locations, each assigned with a uniformly random parameter for each respective potential model: $m_j \sim U(0,1)$ for gravity, $q_i \sim U(-1,1)$ for electrostatic potential, and $k_{ij} \sim U(0,1)$ for elastic potential. We first observe that, in the case of gravity and electro-static models, the system converges to an equilibrium more quickly with the local search method, in terms of the total distance the particles traveled, than the global search, yet with the penalty of $\Phi(t)$ not reaching the level of what global method achieves. This is an expected behavior of particles moving in a hop-by-hop manner being caught in local minima formed by these two potential energy fields. With the elastic energy model, however, there is no clear difference between the two methods. It is because the potential field generated by the sum of individual elastic potential function ($k_{ij}d_{i,j}(t)^2$) is actually a quadratic, convex function, and hence there is no local minimum that the particles get caught at each step of movement.

Size	Model	Energy	Move Distance	Iterations
n=10 p=10	Gravity	0.296	0.277	2.235
	ElectroStatic	0.873	0.405	2.102
	Elastic (Spring)	1.000	0.941	2.932
n=50 p=50	Gravity	0.061	0.048	3.485
	ElectroStatic	0.822	0.074	1.958
	Elastic (Spring)	1.000	0.973	12.065

Figure 3. Local search vs Global search with different potential energy models: the competitive ratios of (i) total reduction of the system energy, (ii) total distance the particles have moved until convergence, and (iii) the total number of iterations until convergence.

To investigate deeper the above observation, we perform 100 simulations for each scenario, and Figure IV presents the average values of the competitive ratio of local search to global one, in terms of (i) the *reduction* of system potential energy in equilibrium from the initial one, (ii) the total number of distances the particles moved, and (iii) total number of iterations till convergence, where an iteration means a round of moving (or not moving) each and every particle in the system once. The results validate our qualitative observation above. There is one noteworthy finding, however: the local search method is more competitive to the global one with electro-static potential than the gravity, i.e., the local search within electro-static potential fields tends to result in almost as much reduction of the system potential energy as what global search achieves, with only a small fraction of the total movement distance (especially with larger system). We suspect that this is because particles with the opposite polarity in electro-static model get attracted to each other, but once they are co-located or placed nearby, their combined force becomes neutralized (weaker), hence having the effect of balancing out the placements of the particles. In contrast, in the gravitational field (even with particles with negative mass), a cohesion of the particles due to attractive force makes them even stronger attractive matter, resulting in the formation of deeper and deeper potential wells as the particles move about.

Our last experimental result explores a scenario in which attractive and repulsive forces are assigned to the elements



(a) Initial, random placement (b) Rc-C, Rd-D attraction (c) Rc-C, Rd-D attraction + C-C, D-D repulsion + C-D attraction(k_e=0.5) (d) Rc-C, Rd-D attraction + C-C, D-D repulsion + C-D attraction(k_e=1.0)

Figure 4. Distributed placement of codes and data due to electro-static potential minimization on 1-D graph of 20 nodes and 20 particles

of ACAD systems using electro-static potential model, as described in Section III-C. Figure 4 shows the resulting placement of the particles on 1-D graph of 20 nodes in a simulation run, C (Codes) and D (Data), while Rc (Compute Resource) and Rd (Data Resource) are modeled as immovable particles yet exerting attractive forces to C and D, respectively. Figure 4(a) is an initial placements of C and D evenly placed out in the graph, and Figure 4(b) shows the result of exerting attractive forces only for Rc-C and Rd-D interaction, which exhibits C and D are well load-balanced across their respective resources. Figures 4(c) and 4(d) both show the placements when the additional forces between codes and data are introduced (C-D, C-C, and D-D), with the difference being the strength of C-D attractive force, relative to the other forces, is weighted down by half in Figure 4(c) by assigning smaller k_e (Coulomb constant). We observe, as C-D attraction becomes stronger, some portion of codes and data "leave" their respective silos and rendezvous each other at some mid-points, indicating the codes being executed against data in an agile manner. While being a qualitative observation, this indicates the physical models and their parameter settings can act as an effective control-knob for allocating resources and placing codes and data in a distributed manner.

V. Discussion and Future Work

These are all parts of our on-going efforts to find and apply fundamental models in agile composition of the computing tasks and data in distributed edge systems, and there are several interesting research directions from here. As discussed above, finite temperature simulations, especially Metropolis Monte Carlo (MMC), would aid in expanding the search space. Simulated annealing (SA), based on MMC, could be used to obtain the global optimum solution, while grand-canonical MMC could be used to vary the number of particles, simulating the replication of data and code resources. Alternatively, models from other, non-physics disciplines, such as evolutionary dynamics [7] may prove useful. Another interesting question is whether some "local" dynamics and interactions of the objects can help improve the performance in localized regions but do not impact long-term, global stability of the system. Also, more practical questions beyond the theoretical modeling concern the applicability of the models in real systems; e.g., how the force field information can get distributed to all objects in the system? How the "movement" of objects shall be realized, as the real, physical movement or only simulated ones? We plan to investigate these issues in the future.

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

References

[1] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[2] D. Saad, C. H. Yeung, G. Rodolakis, D. Syrivelis, I. Koutsopoulos, L. Tassiulas, R. Urbanke, P. Giaccone, and E. Leonardi, "Physics-inspired methods for networking and communications," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 144–151, 2014.

[3] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. Oxford, UK: Calrendon Press, 1987.

[4] D. Frenkel and B. Smit, *Understanding Molecular Simulation*, 2nd ed. Orlando, FL: Academic Press, Inc., 2002.

[5] M. P. Allen *et al.*, "Introduction to molecular dynamics simulation," *Computational soft matter: from synthetic polymers to proteins*, vol. 23, pp. 1–28, 2004.

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

[7] M. A. Nowak, *Evolutionary dynamics*. Harvard University Press, 2006.