

Software System for Optimization of Metallic Nanostructures

Rossen Mikhov
0000-0001-7186-0110
Institute of Information and
Communication Technologies,
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 2,
1113 Sofia, Bulgaria

Vladimir Myasnichenko
0000-0003-4355-7055
Tver State University
33, Zhelyabova Str.,
170100 Tver, Russia
Email: viplabs@ya.ru

Leoneed Kirilov^{1,2}
0000-0001-8424-9070
¹ Institute of Information and
Communication Technologies,
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 2,
1113 Sofia, Bulgaria
Email: l_kirilov_8@abv.bg
² Centre of Excellence in
Informatics and Information and
Communication Technologies,
Sofia, Bulgaria

Nickolay Sdobnyakov
0000-0003-1749-0336
Tver State University
33, Zhelyabova Str.,
170100 Tver, Russia

Stefka Fidanova^{1,2}
0000-0002-8484-5849
¹ Institute of Information and Communication
Technologies,
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 2,
1113 Sofia, Bulgaria
² Centre of Excellence in Informatics and Information
and Communication Technologies, Sofia, Bulgaria

Abstract—A software system for numerical optimization of metallic nanostructures using Monte Carlo methods is presented in this paper. It is designed for high performance and flexibility, allowing the construction of complex experiment plans combining multiple Monte Carlo algorithms and for systematic variation of their parameters. The system runs on Linux and Windows operating systems, and is compatible with a wide range of existing software packages using the standard XYZ file format for atomic configurations. It has been used for modeling and optimization of various metallic and bimetallic nanostructures from 150 to 16,000 atoms, including nanoparticles, nanowires, nanofilms and nanocages.

Index Terms—nanostructures, simulation, software systems, Monte Carlo methods, optimization.

I. INTRODUCTION

THE problem of optimizing the structure of nanoparticles and nanostructured materials is closely related to their stability, especially when it comes to nanoclusters containing only a few hundred atoms [1]. At the same time, optimization algorithms for various types of nanoobjects have recently been actively tested using the Monte Carlo method

[2]–[5]. This approach is interesting, including from a prognostic point of view for describing the manifestations of instability of nanoscale objects [6]. At the same time, the problem of structural optimization can be complicated by changes in external factors, primarily temperature [7]. Thus, despite some progress in the development of approaches to optimizing the structure of nanoparticles and nanostructured materials, a number of open questions remain, including those related to the optimization algorithm itself.

This paper presents a software system for Monte Carlo optimization of metallic nanostructures. We formulate the primary requirements for such a system, propose a software architecture suitable for meeting those requirements, and describe the components of the system in the way they have been implemented. A description is provided of the computational model on which the implemented Monte Carlo algorithms work, and an illustrative example is given using the software.

II. REQUIREMENTS FOR THE SOFTWARE SYSTEM

For experimenting with Monte Carlo optimization of various types of metallic and bimetallic nanostructures, the software system should implement a number of Monte Carlo algorithms, providing flexibility in the manner they can be combined, and allowing systematic variation of their parameters. It should be easy to use not only for investigating the

This work was partially supported by the Centre of Excellence in Informatics and ICT under the Grant No BG16RFPR002-1.014-0018-C01, financed by the Research, Innovation and Digitalization for Smart Transformation Programme 2021-2027 and cofinanced by the European Union and by the IICT-BAS project No210386 “Models and methods for decision support and applications” with Virtech LTD, Sofia.

properties of the studied nanoparticles, but also for investigating the behavior of the algorithms themselves.

We formulated the primary requirements for the system as follows:

- It must be optimized for x86-64 processors, achieving performance sufficient to execute billions of Monte Carlo iterations within minutes, thereby enabling the algorithms to reach good solutions for nanoparticles with several hundred to several thousand atoms on standard and widely available hardware.
- It must use the standard XYZ data format for input and output of atomic configurations, ensuring compatibility with a wide range of existing software packages, such as those for 3D visualization, analysis of the results, etc.
- It must be compatible with Linux and Windows operating systems.
- It must support the construction of complex plans for numerical experiments that combine multiple processing steps, various algorithmic variants, parameter variations, output of intermediate data for animated 3D visualization, performance statistics, etc.
- It must be able to synchronize the execution of the experiment plan across multiple processor cores on a single computer or across multiple computers with different operating systems, while remaining robust and capable of continuing normal operation after unexpected interruptions (e.g., power failures or system restarts).
- It must provide a high level of operational flexibility, including the ability to modify the experiment plan after the experiment has already been started, in a way that results already obtained remain usable for the updated plan.

III. SOFTWARE ARCHITECTURE

The software architecture consists of three main components: Computational core, a template for the Experiment Master Plan, and auxiliary functionalities for analysis of the results, testing, and visualization (Fig. 1).

The Computational core is concentrated in a single executable command written in the C programming language [8]. It incorporates all Monte Carlo algorithms and their variations. Each execution of this command corresponds to a single run of a single algorithm.

The Experiment Master Plan defines what nanoparticles are to be studied, what combinations of algorithms are to be used, whether parameters will be varied, what results are to be collected, where they will be stored, and how they will be aggregated for visualization. Essentially, the Experiment Master Plan consists of one Makefile with a specific structure. However, here we do not just use a standard Make command, but instead we combine some of the lesser-known advanced features of GNU Make [9] with a number of cus-

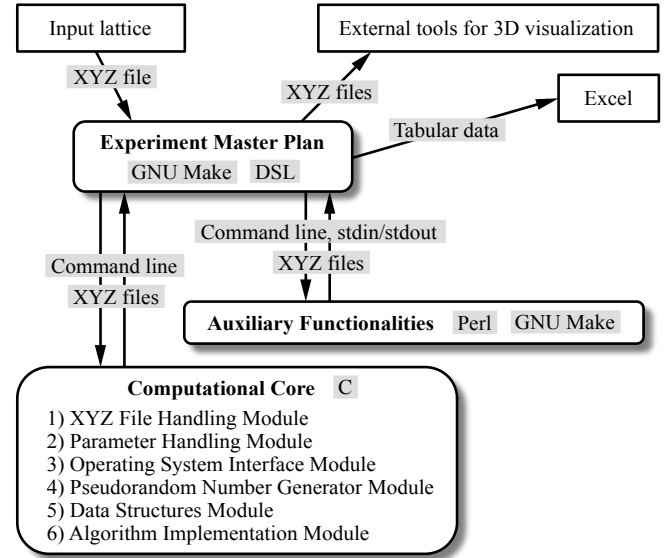


Fig 1. Main components of the software architecture

tom extensions developed specifically for this system. In this way Make is transformed from a cumbersome tool into a flexible and expressive domain-specific language (DSL), suitable for achieving the objectives outlined in Section II.

The auxiliary functionalities consist of a library of GNU Make functions implemented using Perl [10] scripts, which provide the tools necessary for expressing the Experiment Master Plan. Additionally, an automated regression testing system for the implementation of the algorithms of the Computational core is also included.

Graph visualization is done in Excel, using data automatically extracted in real time and formatted according to a table schema defined in the Experiment Master Plan. 3D visualization of atomic configurations is performed separately using standard widely available software packages capable of reading XYZ files.

This architecture isolates the core computational workload into a single command with a clearly defined interface, enabling careful code optimization for performance. At the same time, the experimental workflow of a whole experiment is centralized in a single Makefile, supported by an extensive library of specialized functions, ensuring both flexibility and transparency in experiment design and documentation. This architecture is also easily extensible, allowing for future enhancements as the need arises, such as graphical or web-based user interfaces built upon the existing components.

A. Computational Core

The Computational core is implemented as a single command, each execution of which corresponds to a single run of a single algorithm. The core consists of six modules:

1) XYZ File Handling Module

This module provides all necessary functions for reading and writing data in the *de facto* standard XYZ file format for

atomic configurations. The format is structured as follows (Fig. 2):

- The first line contains a single integer n , which represents the number of nodes in the lattice.
- The second line has no standardized meaning, and is typically ignored when read by other software. For the present system, it contains a long list of variables and their values, used for input parameters, output results (e.g., energy, temperature; everything other than the (x, y, z) coordinates of the atoms), and miscellaneous internal state.
- Each of the next n lines represents a node in the lattice and contains the following columns, separated by spaces or tab characters: the chemical element (or the letter n , if the node is empty), x , y , z coordinates. Depending on the experiment, additional columns may be included, which represent the energy of the given atom, classification of the atom by layer group in the nanoparticle, etc.
- The file may contain multiple configurations (e.g., for animation), in which case the above $n+2$ lines repeat the required number of times.

This module takes special care to store all parameters required to fully reconstruct the system's internal state with sufficient numerical precision, ensuring reproducibility and safe continuation after interruptions.

The module also reads parameters for the potential energy function from a separate file format.

2) Parameter Handling Module

This module defines all parameters used by the system, together with the rules for their manipulation. Parameters can be read both from the command line and from the input XYZ file, and are written to output XYZ files. They may be input or output parameters. Each parameter has a designated type (e.g., integer, floating point, string), default value, optionally minimum and maximum value, and help text. Example input parameters include the composition (type and proportion of chemical elements in the particle), maximum number of iterations, neighborhood radius, etc. Output parameters include algorithm stop reason, achieved energy, etc. The starting and current state of the random number generator are also stored as parameters, for reproducibility of the results. The system uses approximately 50 parameters in total.

The parameter system is designed in such a way that any generated XYZ file can serve as input for a subsequent run of the same or a different algorithm, with a simple modification of the command line. This, besides being useful when defining an Experiment Master Plan, also helps for the seamless continuation after intentional or unintentional interruptions.

3) Operating System Interface Module

This module contains low-level functions with platform-specific implementations for Linux and Windows.

```

3000
#Algo=Diffusion Task=AuAg-3000_4500-fcc_080M_Tprev1500_T1500_seed25
#Energy=-9583.0102125318499 MinEnergy=-9583.0102125318499
Temper=551.57657207140539 IterCnt=50618260 IterMax=80000000
#StopReason=IterMax JumpMax=80000000 JumpCnt=50618260
PbbyJumpCnt=25311619 EqlbrIterMax=80000000 EqlbrIterCnt=0
TemperChangelterMax=10000 TemperChangelterCnt=8260
SnapshotIterMax=80000000 SnapshotIterCnt=50618260 #Nodes=3004
Num=3000 Composition=AuAg Rneigh=5.26 #Rcut=6.7823411887046792
TemperDelta=-1.8739842480310037e-05 #ZoomX=1.002 #ZoomY=1
#ZoomZ=1 ZoomMin=0.8 ZoomMax=1.2 SaveNodes=0
RandState=2D43C0D10795E7100E4D586B2CFAB6D7
#InitialRandState=5250B2EA33F5F116E134B987F06DA224
Au -5.8165665933600001 -1.9214831400000001 -19.320616619999999
Au -9.8402655085199981 -17.709002340000001 11.800856339999999
Au 1.8900739026000002 -0.0004763400000000001 -17.068808520000001
Au -1.9485816044400002 21.585799979999997 7.8422465399999997
Au 7.9086129746400005 19.785529560000001 -11.86733892
[...]
Ag 3.7796173664399997 -11.328081239999999 -11.32314648
Ag 1.9835312842800001 13.8258042 19.75329348
Ag 3.99379606884 -23.70382386 -3.9047772599999999
Ag -7.6161807576000005 -11.41884288 11.415396300000001
Ag -9.7092440686800003 -15.49355826 -9.6807108599999996
[...]
    
```

Fig 2. Example XYZ file. “⋮” means continuation of the previous line

4) Pseudorandom Number Generator (PRNG) Module

As Monte Carlo simulations use an enormous number of pseudorandom numbers, a suitable choice of PRNG is essential. Default PRNGs typically have a number of disadvantages, so we specifically chose the xoroshiro128+ algorithm [11], which ensures high statistical quality for floating-point random number generation. The algorithm has a long period $2^{128}-1$, compact internal state (only 32 hex digits, which is convenient for use in XYZ files and on the command line), and has code that is especially optimized for x86/x86-64 processors.

5) Data Structures Module.

Defines the data structures and associated operations required for the implementation of the algorithms.

6) Algorithm Implementation Module

Implements the computational model and algorithms described in Section IV and their variants, with a strong emphasis on performance for x86-64 processors. To this end, the code is organized in a way that enables the compiler to analyze it as much as possible in order to apply built-in optimizations. The most critical code sections are implemented in multiple ways, then profiled on representative examples in real conditions, to select the most efficient coding approach.

B. Experiment Master Plan

Each experimental study typically requires hundreds of algorithm runs under varying conditions. In the proposed architecture, the entire experiment is defined in a single Makefile.

The XYZ files are stored in a small number of directories (often one) on a shared filesystem. All the information about what the XYZ represents in the context of the Experiment Master Plan is encoded in the file name. The file name is parsed by the system into a list of variables that can be used to construct command-line parameters for the Computational core, in case the file will subsequently be used as input to some algorithm.

To facilitate expressing complex experiment plans in a concise and declarative manner, the variables and parameters can be combined using a custom domain-specific language (DSL), which supports arithmetic operators and other functions, type checking, regular expressions, and other features. This approach enables a high degree of flexibility to plan studies of different kinds of nanoparticles, or studies of the behavior of the algorithms themselves.

The developed DSL also functions as a query language over the results, enabling the extraction of specific information from the potentially thousands of generated XYZ files, and aggregation of data in tabular form for real-time visualization, according to a user-defined table schema.

The use of static descriptive file names in one shared directory, and GNU Make for generating those files, brings all classic advantages of GNU Make, such as the possibility of synchronized execution using several processor cores on one or several computers, possibly with different operating systems. The system has features for robustness that allow it to continue normal operation after unexpected interruptions, and features for operational flexibility that allow modifying the Master Plan after the experiment has already been started without loss of work. The developed DSL and other extensions compensate for many of the disadvantages of the Makefile language, specifically for use in experiments with nanoparticles.

C. Auxiliary Functionalities for Analysis and Testing

The auxiliary functionalities consist of a library of GNU Make functions, implemented using Perl scripts. They support the DSL described above, and provide additional tools for experiment management, including:

- various commands for XYZ file processing;
- progress visualization with color;
- logging;
- performance statistics.

An automated regression testing system for the Computational core is also included.

A graphical user interface is not provided, because our past usage of the system has always been very dynamic, with each experiment having a unique element in its design that evolves even during the process of its execution. It would be difficult to warp such usage in some standardized collection of buttons and menus. Instead, our system supports convenient batch execution of trials. Should the system ever become popular among non-technical users, the architecture allows for the addition of graphical or web-based interfaces on top of the existing components.

IV. COMPUTATIONAL MODEL AND ALGORITHMS

Two basic Monte Carlo algorithms, developed by the authors, are currently provided:

- a wide-lattice simulated annealing algorithm [12];
- an algorithm for the simulation of diffusion [13].

The system works with a pre-prepared model of the nanostructure. For example, in the case of a bimetallic

nanoalloy, the total number of atoms is known, as well as the number of atoms of each metal separately (proportion). The atoms are arranged on a predefined lattice consisting of empty nodes. The first goal is to find a configuration of atoms on this lattice so that the resulting nanoparticle is stable—i.e., to minimize the potential energy of the system.

The Cleri–Rosato many-body tight-binding (TB) potential [14] is used to calculate the interaction between the atoms. The total potential energy for the entire system has the following form:

$$E = \sum_i \sum_{j \neq i} A_{ab} \exp(-p_{ab}(\frac{r_{ij}}{r_{0,ab}} - 1)) - \sum_i \sqrt{\sum_{j \neq i} \xi_{ab}^2 \exp(-2q_{ab}(\frac{r_{ij}}{r_{0,ab}} - 1))}, \quad (1)$$

where i enumerates all atoms in the system; j enumerates all atoms other than i , but at a distance from atom i less than the predefined global parameter R_{cut} ; a and b denote the chemical element of atoms i and j ; r_{ij} is the distance between the atoms; $r_{0,ab}$, A_{ab} , p_{ab} , ξ_{ab} , and q_{ab} are constants specific to the chemical elements under consideration.

The modeled nanoalloy can have nanoscale dimensions along some of the coordinate axes, and macroscale dimensions along the remaining axes. A distinction is made between 3 cases:

- nanoparticle, where all three axes are nanoscale;
- nanowire, where one of the axes is macroscale;
- nanofilm, where two of the axes are macroscale.

In case of a nanowire or a nanofilm, the model is analogous to that for a nanoparticle, but using periodic boundary conditions. This means that the used lattice is nanoscale along all three axes, but a periodic window is introduced along the axes where macroscale dimensions are modeled. The distance between two atoms is thus defined as follows:

$$r_{ij} = \sqrt{|\Delta x_{ij}|^2 + |\Delta y_{ij}|^2 + |\Delta z_{ij}|^2} \quad (2)$$

$$|\Delta x_{ij}| = \min(|x_i - x_j|, L_x - |x_i - x_j|), \quad (3)$$

where x_i and x_j are the x -coordinates of atoms i and j inside the periodic cell, and L_x is the x -size of the periodic cell. If macroscale dimensions are modeled along the y and z axes, then $|\Delta y_{ij}|$ and $|\Delta z_{ij}|$ are also calculated in a similar way. The non-periodic case corresponds to $L_x = \infty$.

The present model, as well as the algorithms for it, are suitable for the optimization of bimetallic nanoparticles with sizes between 100 and several tens of thousands of atoms.

V. ILLUSTRATIVE EXAMPLE

The system has been used for modeling various metallic and bimetallic nanostructures from 150 to 16,000 atoms. In this section, we give an example for using it to optimize six gold–silver nanocages of 3000 atoms each [15].

The Experiment Master Plan for this example is partially illustrated in Fig. 3. It combines both the wide-lattice simu-

lated annealing algorithm and the diffusion algorithm in a hybrid optimization approach consisting of four steps:

Step 1. Parameter setting for the wide-lattice Monte Carlo algorithm. The algorithm is run 30 times from random initial configurations for 40×10^6 iterations in each run, to determine the optimal initial temperature.

Step 2. Fixing of the geometric shape. The wide-lattice algorithm is run again, 30 times from random initial configurations, but this time for 1000×10^6 iterations in each run, using the best values from *Step 1* for the initial temperature.

Step 3. Parameter setting for the diffusion Monte Carlo algorithm. The algorithm is run 30 times on the results of *Step 2*, for 8×10^6 iterations in each run, to determine the optimal initial temperature for diffusion.

Step 4. Diffusion. The diffusion algorithm is run again, 30 times on the results of *Step 2*, for 80×10^6 iterations in each trial, using the best values from *Step 3* for the initial temperature.

The parameters of the Computational core are set as appropriate for each invocation. The results are queried and aggregated in tabular form for analysis and visualization (the table schema is not shown in the figure). A sample achieved configuration of one of the investigated nanocages is displayed in Fig. 4.

VI. CONCLUSION

A software system for numerical optimization of metallic nanostructures using Monte Carlo methods has been presented. The architecture consists of a Computational core, designed for high performance, a template for the Experiment Master Plan, designed for flexibility, and auxiliary functionalities for analysis of the results, testing, and visualization. The system is suitable both for investigating the properties of the studied nanostructures, and for investigating the behavior of the algorithms. The system runs on Linux and Windows operating systems, and is compatible with external tools using the standard XYZ file format for atomic configurations. It has been used for modeling and optimization of various metallic and bimetallic nanostructures from 150 to 16,000 atoms, including nanoparticles, nanowires, nanofilms and nanocages.

REFERENCES

- [1] V. S. Myasnichenko, P. M. Ershov, N. Yu. Sdobnyakov, R. Mikhov, and L. Kirilov, "Thermal stability of copper clusters with the size of 100 – 150 atoms," *Physical and chemical aspects of the study of clusters, nanostructures and nanomaterials*, vol. 9, pp. 330–336, 2017, <https://doi.org/10.26456/pcascnn/2017.9.330>
- [2] U. Gahn, "Atom distributions in face-centered cubic solid solutions from Monte Carlo computer simulations," *Journal of Physics and Chemistry of Solids*, vol. 43, no. 10, pp. 977–989, 1982, [https://doi.org/10.1016/0022-3697\(82\)90114-7](https://doi.org/10.1016/0022-3697(82)90114-7)
- [3] M. C. Giménez and W. Schmicker, "Monte Carlo simulation of nanowires of different metals and two-metal alloys," *The Journal of Chemical Physics*, vol. 134, no. 6, pp. 064707-1–064707-6, 2011, <https://doi.org/10.1063/1.3549900>

```
# All lists undergo Bash-like brace expansion.
S ?= 00..29
STEP1 ?= \
    AuAg--3000-4500_fcc_0040M_T1000,1500,2000,2500}_seed{$S} \
    AuAg--3000-4500_ico_0040M_T2000,2500,3000}_seed{$S} \
    Au3Ag--3000-4500_fcc_0040M_T1000,1500,2000}_seed{$S} \
    Au3Ag--3000-4500_ico_0040M_T2500,3000,4000}_seed{$S} \
    AuAg3--3000-4500_fcc_0040M_T1500,2000,2500}_seed{$S} \
    AuAg3--3000-4500_ico_0040M_T2000,2500,3000}_seed{$S}
STEP2 ?= \
    AuAg--3004-4500_fcc_1000M_T1500_seed{$S} \
    AuAg--3004-4500_ico_1000M_T2500_seed{$S} \
    Au3Ag--3004-4500_fcc_1000M_T1500_seed{$S} \
    Au3Ag--3004-4500_ico_1000M_T3000_seed{$S} \
    AuAg3--3004-4500_fcc_1000M_T2000_seed{$S} \
    AuAg3--3004-4500_ico_1000M_T2500_seed{$S}
STEP3 ?= \
    AuAg--3000-3004_4500-fcc_008M_Tprev1500_T1000,1500,2000}_seed{$S} \
    AuAg--3000-3004_4500-ico_008M_Tprev2500_T1500,2000,2500,3000}_
    _seed{$S} \
    Au3Ag--3000-3004_4500-fcc_008M_Tprev1500_T1000,1500,2000}_seed{$S} \
    Au3Ag--3000-3004_4500-ico_008M_Tprev3000_T1000,1500,2000,2500,3000}_
    _seed{$S} \
    AuAg3--3000-3004_4500-fcc_008M_Tprev2000_T1000,1500,2000}_seed{$S} \
    AuAg3--3000-3004_4500-ico_008M_Tprev2500_T1500,2000,2500}_seed{$S}
STEP4 ?= \
    AuAg--3000-3004_4500-fcc_080M_Tprev1500_T1500_seed{$S} \
    AuAg--3000-3004_4500-ico_080M_Tprev2500_T2500_seed{$S} \
    Au3Ag--3000-3004_4500-fcc_080M_Tprev1500_T1500_seed{$S} \
    Au3Ag--3000-3004_4500-ico_080M_Tprev3000_T2000_seed{$S} \
    AuAg3--3000-3004_4500-fcc_080M_Tprev2000_T1500_seed{$S} \
    AuAg3--3000-3004_4500-ico_080M_Tprev2500_T2000_seed{$S}
SRO ?= \
    AuAg--3000-3004_4500-fcc_080M_Tprev1500_T1500_seed25.ssh \
    AuAg--3000-3004_4500-ico_080M_Tprev2500_T2500_seed11.ssh \
    Au3Ag--3000-3004_4500-fcc_080M_Tprev1500_T1500_seed04.ssh \
    Au3Ag--3000-3004_4500-ico_080M_Tprev3000_T2000_seed21.ssh \
    AuAg3--3000-3004_4500-fcc_080M_Tprev2000_T1500_seed28.ssh \
    AuAg3--3000-3004_4500-ico_080M_Tprev2500_T2000_seed01.ssh

TASKS ?= $(STEP1) $(STEP2) $(STEP3) $(STEP4) $(SRO)

taskPtn = »Composition:SrtNat?Srt?Nat?«--»Num:Nat«--»nodes:Nat«
    _ltnodes:Nat?«ltnSep:Ch??«ltnShp:Nm«_yt:Dbk«M»tprevSep:Ch*?«
    _tprev:Nat?«_T»Temper:Dbk«_seed»seed:Nat«ssht:Ch*«
parmsCode = inputClsc=input_--»nodes«_ltnShp.xyz \
    inputDfsn=trialsAll»Composition«--»nodes«_ltnShp«_ltnShp«_1000M_
    _T»tprev«_seed»seed«_xyz \
    InputFile=ltnSep?inputDfsn?inputClsc«_Rneigh»ltnSep?5.26|3.3« \
    IterMax»rnd(it*1e6)«:(it<=4000)«_JumpMax»IterMax« \
    EqIbIterMax»IterMax«_TemperChangeIterMax=10000 \
    TemperDelta»dbl(0-(Temper-1)/(IterMax-1e3))« \
    SnapshotIterMax»ssht?rnd(rnd(0-100)/(TemperDelta*TemperChangeIterMax))*
    _TemperChangeIterMax)IterMax« \
    SnapshotIterCnt»ssht?rnd(SnapshotIterMax-TemperChangeIterMax)|0« \
    MinEnergy=INF \
    IterCnt=0 JumpCnt=0 PbtyJumpCnt=0 EqIbIterCnt=0 TemperChangeIterCnt=0 \
    zoom»bool(it>=80)« \
    ZoomMin»zoom?it==320?0.99|0.8|1«_ZoomMax»zoom?it==320?1.01|1.2|1« \
    tgt=$$$$@ nul=$(NulNativ) SnapshotFile»ssht?tgt|nul« \
    MinEnergyFile»ssht?nul?zoom?nul|tgt«_ZoomFile»ssht?nul|tgt«_SaveNodes=0
```

Fig 3. Fragment of the Experiment Master Plan for the given example. The “\” symbol denotes continuation of the previous line.

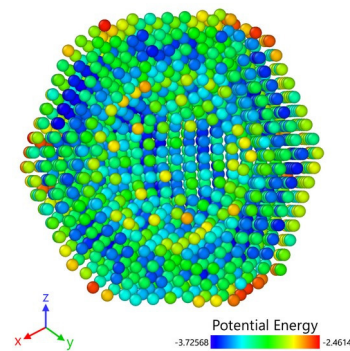


Fig 4. An achieved atomic configuration of the AuAg ico nanocage

- [4] M. Müller and K. Albe, “Kinetic lattice Monte-Carlo simulations on the ordering kinetics of free and supported FePt L10-nanoparticles,” *Beilstein Journal of Nanotechnology*, vol. 2, pp. 40–46, 2011, <https://doi.org/10.3762/bjnano.2.5>
- [5] V. Myasnichenko, N. Sdobnyakov, L. Kirilov, R. Mikhov, and S. Fidanova, “Monte Carlo approach for modeling and optimization of one-dimensional bimetallic nanostructures,” in *Numerical Methods and Applications: NMA 2018*, G. Nikolov, N. Kolkovska, and K. Georgiev, Eds. Springer, 2019, pp. 133–141, https://doi.org/10.1007/978-3-030-10692-8_15
- [6] V. Myasnichenko, N. Sdobnyakov, L. Kirilov, R. Mikhov, and S. Fidanova, “Structural instability of gold and bimetallic nanowires using Monte Carlo simulation,” in *Recent Advances in Computational Optimization*, S. Fidanova, Ed. Springer, 2020, pp. 133–145, https://doi.org/10.1007/978-3-030-22723-4_9
- [7] R. Mikhov, V. Myasnichenko, S. Fidanova, L. Kirilov, and N. Sdobnyakov, “Influence of the temperature on simulated annealing method for metal nanoparticle structures optimization,” in *Advanced Computing in Industrial Mathematics: BGSIAM 2018*, I. Georgiev, H. Kostadinov, and E. Lilkova, Eds. Springer, 2021, pp. 278–290, https://doi.org/10.1007/978-3-030-71616-5_25
- [8] *Information technology — Programming languages — C*, ISO/IEC Standard 9899, ed. 3, 2011.
- [9] *GNU Make Manual*, Free Software Foundation, Feb. 2023, <https://www.gnu.org/software/make/manual/>
- [10] *Perl Documentation*, Perl 5 Porters, Apr. 2026, <https://perldoc.perl.org>
- [11] D. Blackman and S. Vigna, “Scrambled linear pseudorandom number generators,” *ACM Transactions on Mathematical Software*, vol. 47, no. 4, pp. 36:1–36:32, 2021, <https://doi.org/10.1145/3460772>
- [12] V. Myasnichenko, L. Kirilov, R. Mikhov, S. Fidanova, and N. Sdobnyakov, “Simulated annealing method for metal nanoparticle structures optimization,” in *Advanced Computing in Industrial Mathematics: BGSIAM 2017*, K. Georgiev, M. Todorov, and I. Georgiev, Eds. Springer, 2019, pp. 277–289, https://doi.org/10.1007/978-3-319-97277-0_23
- [13] R. Mikhov, V. Myasnichenko, L. Kirilov, N. Sdobnyakov, P. Matrenin, *et al.*, “On the problem of bimetallic nanostructures optimization: an extended two-stage Monte Carlo approach,” in *Recent Advances in Computational Optimization: WCO 2020*, S. Fidanova, Ed. Springer, 2022, pp. 235–250, https://doi.org/10.1007/978-3-030-82397-9_12
- [14] F. Cleri and V. Rosato, “Tight-binding potentials for transition metals and alloys,” *Physical Review B*, vol. 48, no. 1, pp. 22–33, 1993, <https://doi.org/10.1103/PhysRevB.48.22>
- [15] V. Myasnichenko, R. Mikhov, N. Sdobnykov, L. Kirilov, and A. Bazulev, “Lattice Monte Carlo simulation of atomic ordering in gold-silver nanocages,” *WSEAS Transactions on Electronics*, vol. 16, pp. 157–167, 2025, <https://doi.org/10.37394/232017.2025.16.16>