DEVELOPMENT AND ANALYSIS OF THE PARALLEL ANT COLONY OPTIMIZATION ALGORITHM FOR SOLVING THE PROTEIN TERTIARY STRUCTURE PREDICTION PROBLEM

Leonid Hulianytskyi, Vitalina Rudyk

Abstract: *Parallel ant colony optimization algorithm for solving protein tertiary structure prediction problem given its amino acid sequence is introduced. The efficiency of developed algorithm is studied and the results of computational experiment on the SCIT supercomputer clusters are discussed.*

Keywords: *combinatorial optimization, protein tertiary structure prediction, ant colony optimization, parallel algorithms, SCIT supercomputer.*

Introduction

Applied science researches often result in the development of models that are convenient to be analyzed with some mathematical methods. Thus at the interfaces between math and biology the branch of computational biology appeared. It includes the set of mathematical models and methods for solving the problems that arise in biology, genetics, pharmacology, medicine. A wide range of statistical methods, data analysis, combinatorial optimization methods are used for that. Combinatorial optimization methods take on special significance in genetics inasmuch as DNA and RNA molecules are encoded as a sequence of genes and the optimization problems on strings arise. They naturally lead to combinatorial optimization problems.

One of computational biology problems is examined, namely the protein tertiary structure prediction problem based on Dill's model [Dill et al, 1995].

Protein Tertiary Structure Prediction Problem

The aim of protein structure prediction is to construct the three-dimensional shape of the molecule (tertiary structure) given its amino acid composition (primary protein structure). Dill's model describes a tertiary structure using some discrete lattice, amino acids that form a protein are placed in its nodes. In order the molecule to remain connected amino acids that are consequent in the primary sequence are placed in the neighboring lattice nodes. All amino acids are labeled as hydrophobic or polar depending on their physical properties. Between closely set hydrophobic amino acids hydrophobic contacts appear. Every structure in a lattice has some nonpositive energy that is equal to the number of hydrophobic contacts in it. It is believed the molecule takes the shape where the minimum of its energy is achieved. Formal statement of this problem appears to be NP-hard combinatorial optimization problem [Berger & Leighton, 1998].

Parallel Ant Colony Optimization Algorithm for Solving the Problem

Ant colony optimization method (ACO) is a approximate scheme developed for solving combinatorial optimization problems that simulates the process of finding the shortest ways to the food by the colony of ants [Dorigo & Stützle, 2004]. The communication between individuals is performed via pheromone trails that contribute to the selection of optimal moving directions. The complexity of the aroused combinatorial optimization problem causes the importance of parallel combinatorial algorithms research and development.

ACO-based algorithms are developed in [Shmygelska & Hoos, 2005; Chu et al, 2005; Fidanova & Lirkov, 2008]. Parallel ACO algorithm we present differs from the ones mentioned above. In particular another pheromone update procedures, heuristic estimations, structure encoding (q-encoding introduced in [Гуляницький &Рудык, 2013]) are used. The diagram of the algorithm is presented on img.1.

The algorithm is designed to be used on multiprocessor computer systems. One host processor is singled out; all others are treated as subordinated, pairwise interactions are made between host and every subordinated processor. The differentiation of the processors into host and subordinated is conventional, the only requirement to the network architecture is the ability to pass the messages from one of the processors to others and back in a short time. The number of processors involved in calculations is equal to the number of the agents in ant population increased by one.

The first step of the algorithm is the initialization of pheromone trails matrix Φ that is performed on the host processor. The number of rows in Φ coincides with the number of code elements in the lattice, the number of columns is equal to the length of the given protein code. Initially every matrix element is set to some value μ_0 >0.

The record (in the sense of minimal energy value) structure among the ones generated during the procedure is saved, it is designated as fold_{rec}.

After the initialization the iterative process is performed. On every iteration the population (the number of agents in population is the parameter of the algorithm) of new structures is generated based on information that is stored in pheromone matrix. After that every structure modifies the pheromone matrix by adding new pheromone trails that intensify the paths that lead to low-energy structures. The creation of the structure starts from the first element and is processed consecutively. The next amino acid position is chosen from the neighboring free (not occupied) ones with the probability

$$P_{i,d} = \frac{[\tau_{i,d}]^{\alpha}[\eta_{i,d}]^{\beta}}{\sum_{l \in D}[\tau_{i,l}]^{\alpha}[\eta_{j,l}]^{\beta}},$$

where D is a set of q-encoding code elements, that reflects the directions to free neighbors of the current (latest occupied) node in the lattice, $\tau_{i,d}$ is the pheromone matrix element that corresponds to the direction d for amino acid number i with regard to amino acid number $i-1$, $\eta_{i,d}$ is a heuristic estimation that depends on the constructed part of the structure, α and β are the parameters of the algorithm that describe the degree of influence of pheromone and heuristic information on the structure composition.

If after some step the construction process reached a deadlock, i.e. we have a situation when all neighbor nodes are occupied, a one-step rollback is performed and deadlock direction is saved to tabu-listnot to be repeated again. The solution construction is performed on subordinated processors that gets pheromone matrix as an input. To reach the effectiveness increase every of the subordinated processors also performs a local search procedure.

Next a pheromone trail $\mathrm{pheromoneTrail}$ is evaluated on every subordinate processor. It is represented as a sequence of pairs (matrix row index – the amount of pheromone) $(d, \Delta_{i,d,c})$, where d is the direction to the i-th amino acid in the structure c, and $\Delta_{i,d,c}$ is the relative quality of the structure c taking into account direction d.

To calculate $\Delta_{i,d,c}$ two ways are proposed. The first one treats $\Delta_{i,d,c}$ as the relative quality of structure that depends on its energy. The second one is developed to take into account the fact that it is required to strengthen only the trails that influence the energy of constructed molecule. The "strength" $\Delta_{i,d,c}$ of a certain direction d in structure c is defined as the number of connections influenced by this direction. Such scheme provides taking into account the suitability of this or that part of a molecule structure.

PheromoneMatrixInitialization(Φ)
fold_{rec} := *null*
While *FinalCondition* **is not reached**
  On c i = 1,..., P
    fold := *FeasibleSolution*(Φ)
    $\mathrm{fold}_{\mathrm{LS}}$:= *LocalSearch*(fold)
    e = *Energy*($\mathrm{fold}_{\mathrm{LS}}$)
    $\mathrm{pheromoneTrail}$ = *CalculatePheromoneTrail*($\mathrm{fold}_{\mathrm{LS}}$)
  On host processor
    For i = 1,..., p
      Receive $\mathrm{e_i}$, $\mathrm{pheromoneTrail_i}$ **from subordinated processor**
      $\Phi := \Phi + \mathrm{pheromoneTrail_i}$
      If *Energy*($\mathrm{fold_{rec}}$) > $\min_i e_i$
        Request structure fold **from the processor** $\arg\min_i e_i$
        $\mathrm{fold_{rec}} := \mathrm{fold}$
      EvaporatePheromone(Φ)
Return fold_{rec}

Img. 1 Parallel algorithm diagram

The constructed sequence of pairs together with the value of the best generated solution energy is then passed to the host processor that analyses and aggregates received information: adds the mentioned amount of pheromone to corresponding matrix elements ($\tau_{i,d} = \tau_{i,d} + \Delta_{i,d,c}{}^{\gamma}$, γ is the parameter of the algorithm),

compares the received energy value with the saved record one. If lower energy value is found additional request to corresponding processor is performed to get the corresponding structure.

To keep the relevance of pheromone matrix pheromone evaporation procedure is used. It simulates the process of pheromone trails evaporation in nature. Every matrix element is multiplied by some positive parameter ρ that is less than one: $\tau_{i,d} = (1-\rho)\tau_{i,d}$, ρ is an evaporation coefficient, it characterizes the fraction of information gathered on previous steps that is kept.

Pheromone evaporation is also performed on the host processor, after that if final condition is not reached the next iteration is executed. In our implementation the process stops if the record solution fold_{rec} has not been changed during a certain number of iterations.

Parallel Algorithm Efficiency Study

Parallelization of the computations does not always lead to computation time decreasing. So it is important to make an analysis of suggested procedure running time. Let's introduce the following designations:

- p - the number of subordinated processors (that is equal to the number of agents in sequential algorithm);
- t_{ini} - estimated initialization time;
- $t_{slaveIter}$ - estimated time consumed by host processor to perform one iteration (it includes solution generation, local search and pheromone trails calculation);
- $t_{phUpdate}$ - estimated time consumed by host processor to update pheromone matrix for one structure;
- t_{vapor} - estimated time consumed on pheromone matrix modification (evaporation);
- t_{exch} - estimated time consumed to exchange the messages between the host processor and one of the subordinated processors (includes pheromone matrix passing from the host processor to the subordinated one and pheromone trail passing back);
- I – the number of algorithm iterations.

Then the estimated parallel algorithm computation time is $t_{parallel} = t_{ini} + (p(\mathrm{t}_{exch} + \mathrm{t}_{phUpdate}) + + \mathrm{t}_{slaveIter} + \mathrm{t}_{vapour}) \times I,$ while the estimated time for sequential algorithm is $t_{seq} = t_{ini} + (p(\mathrm{t}_{phUpdate} + \mathrm{t}_{slaveIter}) + \mathrm{t}_{vapour}) \times I.$

So the acceleration when using parallel algorithm compared to sequential one is

$$k_p = \frac{t_{ini} + (p(\mathrm{t}_{phUpdate} + \mathrm{t}_{slaveIter}) + \mathrm{t}_{vapour}) \times I,}{t_{ini} + (p(\mathrm{t}_{exch} + \mathrm{t}_{phUpdate}) + \mathrm{t}_{slaveIter} + \mathrm{t}_{vapour}) \times I} =$$

$$= 1 + \frac{((p-1)t_{slaveIter} - pt_{exch}) \times I}{t_{ini} + (p(\mathrm{t}_{exch} + \mathrm{t}_{phUpdate}) + \mathrm{t}_{slaveIter} + \mathrm{t}_{vapour}) \times I}$$

If the number of iterations I is big enough the initialization time t_{ini} can be ignored, then

$$k_p \approx 1 + \frac{(p-1)t_{slavelter} - pt_{exch}}{p(\mathrm{t}_{exch} + \mathrm{t}_{phUpdate}) + \mathrm{t}_{slavelter} + \mathrm{t}_{vapour}}.$$

For further analysis let's designate the number of neighbor nodes in the lattice under study as n, and the number of amino acids in the given protein as m. Let's estimate the orders of computation time depending on the values of n and m.

For a start let's estimate the message exchange time. The host processor sends to subordinated processor pheromone matrix consisting of $m \times n$ elements. By turn the subordinated processor sends the results of its computations in a format of sequence of pairs (element index – the amount of pheromone) – $m \times 2$ elements in total. So the size of transmitted messages is $O(m \times n + m \times 2) = O(m \times n)$. Supposing the message transition time is linearly dependant on its size we have $t_{exch} = O(m \times n)$.

During the evaporation step every of $m \times n$ pheromone matrix elements is updated, so $t_{vapour} = O(m \times n)$.

The time consumed to update pheromone matrix by one agent is $t_{phUpdate} = O(m)$, as one element in every row is modified (and the number of rows is m).

It remains to analyze time $t_{slavelter}$. It includes two components – time consumed to generate a feasible solution given the pheromone matrix and pheromone trails calculation time given the molecule structure. Let's start from the second one. The procedure that calculates the amount of pheromone looks through all hydrophobic contacts in the structure (their number is linearly dependent on its length m) and for every such contact some amount of pheromone is added for all the elements between those that form a contact (not more than m elements). So the complexity of the procedure is $O(m^2)$.

Let's estimate the time consumed to generate a feasible solution. To remind, a structure is constructed by sequential supplementing the elements, if at some step supplementing is not possible a rollback is performed. In the worst case the rollback procedure can turn into the brute force in some region; it means the worst-case complexity is $O(\mathrm{e}^m)$. In practice such situations are rare in three-dimensional lattices, but it is worth pointing out that the problem is critical in two-dimension lattices as the deadlock formations are more probable. Theoretical analysis of the average computational complexity turns into determination the number of self-avoiding paths in given lattice; this problem is supposed to be *NP*-hard [Liśkiewicza et al, 2003]. The computation experiment analysis shows that the procedure of solution generation is the most time-consuming, especially on the last algorithm iterations when the pheromone matrix becomes inhomogeneous.

Let's add up the estimations:

$$k_p \approx 1 + \frac{(p-1)O(\mathrm{e}^m) - pO(\mathrm{m}\times\mathrm{n})}{p(O(\mathrm{m}\times\mathrm{n}) + O(\mathrm{m})) + O(\mathrm{e}^m) + O(\mathrm{m}\times\mathrm{n})}.$$

Two conclusions can be carried out of it. First of all if m is big enough the nominator is greater than zero, so the acceleration when using parallel algorithm compared to sequential one is greater than one. Secondly with the growth of m k_p is growing too and converges to p, that proves the practicability of using multiprocessor computer systems for the high-dimension problems.

Conclusions and Line of Further Investigations

To solve the protein tertiary structure prediction problem using multiprocessor computer systems a parallel ACO-based algorithm was developed and implemented. Acceleration estimations for using parallel algorithm compared to sequential one were calculated.

Line of further investigations is comparison of computed theoretical results with the real ones derived from computational experiment results.

Literature

[Berger & Leighton, 1998] B. Berger, T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete // Journal of Computational Biology, 1998, 5(1), pp. 27-40.

[Chu et al, 2005] D. Chu, M. Till, A. Zomaya. Parallel Ant Colony Optimization for 3D Protein Structure Prediction using the HP Lattice Model, 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 2005, 7, pp.193-200.

[Dill et al, 1995] K. Dill, S.Bromberg, K. Yue, K. M. Fiebig, D. Yee, P. Thomas, H. Chan. Principles of protein folding - a perspective from simple exact models // Protein Science, 1995, 4, pp. 561– 602.

[Dorigo & Stützle, 2004] M. Dorigo,T. Stützle. Ant Colony Optimization, Cambridge: MIT Press, MA, 2004, 348 p.

[Fidanova & Lirkov, 2008] S. Fidanova, I. Lirkov. Ant Colony System Approach for Protein Folding, Int. Conf. Multiconference on Computer Science and Information Technology, 2008, pp. 887–891.

[Liśkiewicza et al, 2003] M. Liśkiewicza, M. Ogiharab, S. Todac. The complexity of counting self-avoiding walks in sub graphs of two-dimensional grids and hyper cubes // Theoretical Computer Science, 2003, 304, pp. 129-156.

[Shmygelska & Hoos, 2005] A. Shmygelska, H. Hoos. An ant colony optimization algorithm for the 2D and 3D hydrophobic polar protein folding problem // BMC Bioinformatics, 2005, 6(30), pp. 30–52

[Гуляницький & Рудык, 2013] Л. Гуляницький, В. Рудык. Проблема предсказания структуры протеина: формализация с использованием кватернионов // Кибернетика и системный анализ, 2013, 4, с.130-137.

Authors Information

Leonid Hulianytskyi *– PhD, head of the department in V.M Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, 40 Glushkova ave., Kyiv, Ukraine, 03680; e-mail: leonhul.icyb@gmail.com*

Major fields of scientific research: combinatorial optimization; decision making; mathematical modeling and applications.

Vitalina Rudyk *– junior research assistant in V.M Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, 40 Glushkova ave., Kyiv, Ukraine; e-mail: vitalina.rudyk@gmail.com*

Major fields of scientific research: computational biology, combinatorial optimization, protein structure prediction.