clonal Selection Algorithm Parallellization With MPJExress

by Dr.ayi Purbasari. St.,mt. Turnitin Paper -publikasi 13

Submission date: 19-Oct-2021 12:28PM (UTC+0700)
Submission ID: 1677851998
File name: ._Clonal_Selection_Algorithm_Parallelization_with_MPJExpress.pdf (916.48K)
Word count: 5697
Character count: 33269

2016 8th Computer Science and Electronic Engineering Conference (CEEC)

Conference Proceedings

28th – 30th September 2016

University of Essex, UK

Sponsored by
University of Essex, UK

Technically Co-Sponsored by
IEEE UKRI Computer Chapter

ISBN: 978-1-5090-2050-8

IEEE Catalogue Number: CFP1685L-ART

2016 8th Computer Science and Electronic Engineering Conference (CEEC)

Copyright and Reprint Permission: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limit of U.S. copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. For reprint or republication permission, email to IEEE Copyrights Manager at pubs-permissions@ieee.org.

All rights reserved. Copyright ©2016 by IEEE.

ISBN: 978-1-5090-2050-8

Table of Contents

2016 8th Computer Science and Electronic Engineering Conference (CEEC)

28th – 30th September 2016

University of Essex, UK

Paper Title and Authors	Page
Forward Error Correction with Physical Layer Network Coding in Two-way Relay Free Space Optical Links *Zina Abu-almaalie and Zabih Ghassemlooy (Northumbria University, United Kingdom); Alaa A. S. Al-Rubaie (Ministry of Higher Education, United Kingdom); It Ee Lee (Multimedia University & Northumbria University, Malaysia); Hoa Le Minh (Northumbria University, United Kingdom)*	1
Utilisation of Multipath Phenomenon to Improve the Performance of BCH and RS Codes *Alyaa AL-Barrak (University of Northampton, United Kingdom); Ali Al-Sherbaz (The University of Northampton & School of Science and Technology, United Kingdom); Triantafyllos Kanakis (University of Northampton, United Kingdom); Robin Crockett (The University of Northampton, United Kingdom)*	6
Building Semantic Maps for Blind People to Navigate At Home *Qiang Liu, Ruihao Li, Huosheng Hu and Dongbing Gu (University of Essex, United Kingdom)*	12
Towards Unobtrusive Ambient Sound Monitoring for Smart and Assisted Environments *Naomi Griffiths and Jeannette Chin (Anglia Ruskin University, United Kingdom)*	18
Enabling Wireless Software Defined Networking in Cloud Based Machine-to-Machine Gateway *Bilal R. Al-Kaseem (Brunel University London, United Kingdom); Hamed Saffa Al-Raweshidy (University of Brunel, United Kingdom)*	24
Scalable M2M Routing Protocol for Energy Efficient IoT Wireless Applications *Bilal R. Al-Kaseem (Brunel University London, United Kingdom); Hamed Saffa Al-Raweshidy (University of Brunel, United Kingdom)*	30
Class-Specific Pre-trained Sparse Autoencoders for Learning Effective Features for Document Classification *Maysa Abdulhussain and John Q Gan (University of Essex, United Kingdom)*	36
An Event Detection Approach for Identifying Learning Evidence in Collaborative Virtual Environments *Samah Felemban, Michael Gardner and Vic Callaghan (University of Essex, United Kingdom)*	42
Web-based Visualisation of Head Pose and Facial Expressions Changes: Monitoring Human Activity Using Depth Data *Grigorios Kalliatakis (University of Essex & TEI Crete, United Kingdom); Nikolaos Vidakis (TEI Crete, Greece); Georgios Triantafyllidis (Aalborg University Copenhagen, Denmark)*	48
Focus-Sensitive Dwell Time in EyeBCI: Pilot Study *Giacinto Barresi (Istituto Italiano di Tecnologia, Italy); Jacopo Tessadori (Italian Institute of Technology, Italy); Lucia Schiatti (Istituto Italiano di Tecnologia, Italy); Dario Mazzanti (Fondazione Istituto Italiano di Tecnologia, Italy); Darwin Caldwell (Istituto Italiano di Tecnologia, Italy); Leonardo Mattos (Italian Institute of Technology, Italy)*	54
Refined Data Partitioning for Improved Video Prioritization *Martin Fleury (University of Essex, United Kingdom); Ismail Amin Ali (University of Duhok, Iraq); Sandro Moiron (Instituto de Telecomunicações, United Kingdom); Mohammad Ghanbari (University of Essex, United Kingdom)*	60

Generation and VR Visualization of 3D Point Clouds for Drone Target Validation Assisted by an Operator *Louis-Pierre Bergé (Lab-STICC - Telecom Bretagne - Cranfield University); Nabil Aouf (Cranfield University, United Kingdom); Thierry Duval (Lab-STICC - Telecom Bretagne); Gilles Coppin (Telecom Bretagne, France)*	66
Interpolation of Low Resolution Digital Elevation Models: A Comparison *Hemalatha Kalimuthu (Multimedia University, Malaysia); Tan WN (MMU, Malaysia); Sin Liang Lim and Mohammad Faizal Ahmad Fauzi (Multimedia University, Malaysia)*	71
Auto-tuning Sliding Mode Control for Induction Motor Drives *Fizatul Aini Patakor (Politeknik Merlimau Melaka, Malaysia); Zulhisyam Salleh and Marizan Sulaiman (Universiti Teknikal Malaysia Melaka, Malaysia); Normah Jantan (Politeknik Merlimau Melaka)*	77
Optimization of Fuzzy Logic Based for Vector Control Induction Motor Drives *Zulhisyam Salleh (Universiti Teknikal Malaysia Melaka, Malaysia); Marizan Sulaiman and Rosli Omar (Universiti Teknikal Malaysia Melaka); Fizatul Aini Patakor (Politeknik Merlimau Melaka, Malaysia)*	83
Human Activity Recognition From Automatically Labeled Data in RGB-D Videos *David Jardim and Luís Nunes (Instituto Universitário de Lisboa ISCTE-IUL, Portugal); José Miguel Dias (ISCTE, Portugal)*	89
Routing in Hexagonal Computer Networks: How to Present Paths by Bloom Filters Without False Positives *Gokce Caylak Kayaturan and Alexei Vernitski (University of Essex, United Kingdom)*	95
The Use of Biweight Mid Correlation to Improve Graph Based Portfolio Construction *Patrick Veenstra, Colin Cooper and Steve Phelps (King's College London, United Kingdom)*	101
An Income-Based Real-Time Pricing Algorithm Under Uncertainties in Smart Grid *Sahar Ahmadzadeh and Kun Yang (University of Essex, United Kingdom)*	107
Robot Assisted Evacuation Simulation *Ibraheem Sakour and Huosheng Hu (University of Essex, United Kingdom)*	112
Network Performance Optimization Using Odd and Even Routing Algorithm for Pipeline Network *Siva Kumar Subramaniam (Brunel University London, United Kingdom); Shariq Mahmood Khan (NED University of Engineering & Technology, Pakistan); Rajagopal Nilavalan and Wamadeva Balachandran (Brunel University, United Kingdom)*	118
Quaternion Linear Colour Edge-Sharpening Filter Using Genetic Algorithm *Shagufta Yasmin (University of Essex & Sir Syed University, United Kingdom)*	124
Towards Neuroimaging Real-Time Driving Using Convolutional Neural Networks *Carlos Fernandez Musoles (DeMontfort University, United Kingdom)*	130
Serious Games for Fire and Rescue Training *Warren Viant, Jon Purdy and Jason Wood (University of Hull, United Kingdom)*	136
A Blind Source Separation Approach Based on IVA for Convolutive Speech Mixtures *Tariqullah Jan (University of Engineering and Technology Peshawar, Pakistan); Mohammad Haseeb Zafar (University of Engineering and Technology, Peshawar, Pakistan); Ruhul Khalil (UET Peshawar, Pakistan); Majid Ashraf (University of Engineering and Technology, Peshawar, Pakistan)*	140
Recognizing Arabic Sign Language Gestures Using Depth Sensors and a KSVM Classifier *Miada A. Almasre, M (King AbdulAziz University, Saudi Arabia); Hana Alnuaim (King Abdulaziz University, Saudi Arabia)*	146
Video Frame Extraction for 3D Reconstruction *Louis G Clift and Adrian F. Clark (University of Essex, United Kingdom)*	152
Hybrid Routing Scheme for Vehicular Delay Tolerant Networks *Sayed Fawad Ali Shah (University of Engineering & Technology, Peshawar, Pakistan); Mohammad Haseeb Zafar (University of Engineering and Technology, Peshawar, Pakistan); Ivan Andonovic (University of Strathclyde, United Kingdom); Tariqullah Jan (University of Engineering and Technology Peshawar, Pakistan)*	158
Standard Deviation Based Weighted Clustering Algorithm for Wireless Sensor Networks *Faris AL-Baadani and Sufian Y. Yousef (Anglia Ruskin University, United Kingdom); Laith A Al-Jobouri (University of Essex, United Kingdom); Sourabh Bharti (ABV-Indian Institute of Information Iechnology And Management, Gwalior, India)*	164
Quadrotor Transporting with Cable-Suspended Load Using Iterative Linear Quadratic Regulator	168

(iLQR) Optimal Control *Yaser Alothman and Dongbing Gu (University of Essex, United Kingdom)*	
Rolling Horizon Coevolutionary Planning for Two-Player Video Games *Jialin Liu, Diego Perez Liebana and Simon Lucas (University of Essex, United Kingdom)*	174
FPGA Optimization of Convolution-based 2D Filtering Processor for Image Processing *Gian-Domenico Licciardo, Carmine Cappetta and Luigi Di Benedetto (University of Salerno, Italy)*	180
General Video Game for 2 Players: Framework and Competition *Raluca Gaina, Diego Perez Liebana and Simon Lucas (University of Essex, United Kingdom)*	186
Design and FPGA Implementation of a Real-time Processor for the HDR Conversion of Images and Videos *Gian-Domenico Licciardo, Carmine Cappetta and Luigi Di Benedetto (University of Salerno, Italy)*	192
Energy-Efficient M2M Routing Protocol Based on Tiny-SDWCN with 6LoWPAN *Wasan Twayej (BRUNEL UNIVeRSITY, United Kingdom); Hamed Saffa Al-Raweshidy (University of Brunel, United Kingdom); Muhammad Khan (Brunel University, United Kingdom); Suad El-Geder (Brunel Uneversity, United Kingdom)*	198
Change-Point Cloud DDoS Detection Using Packet Inter-Arrival Time *Opeyemi Osanaiye (University of Cape Town, South Africa); Kim-Kwang Raymond Choo (University of South Australia, Australia); Mqhele E. Dlodlo (University of Cape Town, South Africa)*	204
Reversible Decision Support System: Minimising Cognitive Dissonance in Multi-Criteria Based Complex System Using Fuzzy Analytic Hierarchy Process *Md Mahmudul Hasan, Abu-Hassan Kamal, Khin Lwin and Alamgir Hossain (Anglia Ruskin University, United Kingdom)*	210
Optimisation of the Spatial Discretisation of Myelinated Axon Models *Miguel Capllonch Juan, Florian Kolbl and Francisco Sepulveda (University of Essex, United Kingdom)*	216
Spectral Clustering Using the kNNMST Similarity Graph *Patrick Veenstra, Steve Phelps and Colin Cooper (King's College London, United Kingdom)*	222
Clonal Selection Algorithm Parallelization with MPJExpress *Ayi Purbasari, AP (Universitas Pasundan & Institut Teknologi Bandung, Indonesia)*	228

Clonal Selection Algorithm Parallelization with MPJExpress

Case Study: Traveling Salesman Problem

Ayi Purbasari
Informatics Departement,
Universitas Pasundan, Bandung, Indonesia
pbasari@unpas.ac.id

***Abstract*— This paper exploits the parallelism potential on a Clonal Selection Algorithm (CSA) as a parallel metaheuristic algorithm, due the lack of explanation detail of the stages of designing parallel algorithms. To parallelise population-based algorithms, we need to exploit and define their granularity for each stage; do data or functional partition; and choose the communication model. Using a library for a message-passing model, such as MPJExpress, we define appropriate methods to implement process communication. This research results pseudo-code for the two communication message-passing models, using MPJExpress. We implemented this pseudo-codes using Java Language with a dataset from the Travelling Salesman Problem (TSP). The experiments showed that multicommunication model using `alltogether` method gained better performance that master-slave model that using `send-and receive` method.**

Keyword: Clonal Selection Algorithm, parallelization, parallel design, message passing model, MPJExpress, TSP.

I. INTRODUCTION

In an optimisation problem, we need to find, among many alternatives, a best or good enough solution. In our daily life, there are many instances of optimisation problems. However, if the problems arise on much bigger scales, they are complex and need computer algorithms to solve them. The Travelling Salesman Problem (TSP) in one type of optimisation problem, where there is a salesman who needs to find the shortest possible route to visit each city exactly once and return to his origin city [1]. This problem is a combinatorial optimisation that needs a sophisticated algorithm to solve it.

To solve an optimisation problem, there are exact methods that need no constraints or unlimited resources, such as time and memory. But in the real world there are always limited resources, and this limitation leads to the development of heuristic or metaheuristic approaches as a major field in operation research [2]. These approaches provide optimal or sub-optimal feasible solutions in a reasonable time that significantly reduces the time of the search process. However, the high dimensions of many tasks will always pose problems and result in time-consuming scenarios. Therefore, parallelism is an approach that not only aims to reduce the resolution time but also to improve the quality of the provided solutions. The latter holds since parallel algorithms usually run a different search model with respect to sequential ones [2] [3].

A Clonal Selection Algorithm (CSA) is an algorithm that I inspired by a clonal selection mechanism to provide an immune response [4]. This inspiration results in some algorithms in the class of Artificial Immune System (AIS) algorithms [4]. A CSA (Clonal Selection Algorithm) is one of the population-based heuristic search algorithms. This algorithm has been able to solve combinatorial problems [5] [6], such us the Travelling Salesman Problem (TSP) [7]. Like other population approaches, this algorithm requires a significant amount of computation time. To reduce this, many ideas have attempted to address this problem by adopting a parallel computation paradigm. Dabrowski and Kobale [8] use the parallel-CSA computation for a graph colouring problem. Hongbing et al. [9] apply the CSA parallelism for protein structure prediction using Open-MPI. Purbasari use CSA for multi Travelling Salesman Problem (MTSP) [10].

There are two chief issues in evaluating parallel metaheuristics: how fast solutions can be obtained, and how far they are from the optimum [2] There are two different approaches for analysing metaheuristics: a theoretical aspect (worst-case analysis, average-case analysis) [2] or an experimental analysis. An experimental analysis usually consists of applying the developed algorithms to a collection of problem instances and comparatively reporting the observed solution quality and consumed computational resources (usually time) [2][3]. To do this development of parallel algorithms, we need to analyse and design a parallel algorithm from a current sequential algorithm.

Most papers on parallel metaheuristics do not explain in detail the stages of designing a parallel algorithm itself. They describe the process of transformation of sequential algorithms into parallel algorithms generally. The focus of research is on the end result and the performance of parallelism. In this paper, we will explain the stages and steps in building a parallel algorithm. These detailed stages will be a guide in designing parallel algorithms, so that the resulting one will have a good performance.

This research exploits the available parallelism potential on Clonal Selection Algorithm (CSA). Parallel models are built to refer to the principles and concepts of a parallel computation design from Foster [11].

Systematically, this paper contains: Introduction, The Proposed Method/Algorithm, Research Methods, Results and Discussion, and Conclusion.

II. Research Method

This study started by exploiting the parallelism potentials on a Clonal Selection Algorithm, which lead to several parallel solutions for the TSP problem. We used a parallel design method from Foster that has four distinct stages: partitioning, communication, agglomeration and mapping [11]. The first two stages focus on concurrency and scalability and seek to discover algorithms with these qualities. The third and fourth stages shift to locality and other performance-related issues [11]. After we create a parallel model algorithm, we will evaluate it by using theoretical aspects and then give some guidance to implement it using MPJExpress. Figure 1 below shows the research method:

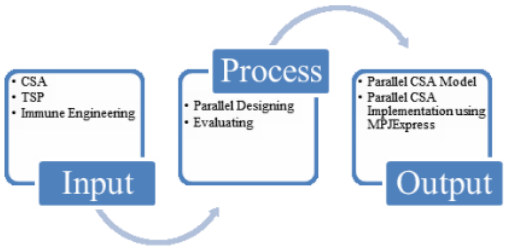

Figure 1 Research Method

This research use CSA, which has many parallelism potentials for the TSP problem. There is a process called immune engineering that is used to map between CSA parameters and processes to solve the TSP problem. The processes consist of parallel design and its evaluation. The output of this research is a parallel CSA model and the guidance to implement it.

Starting with a problem specification, we do parallel design by engaging in partition, determining communication requirements, agglomerating tasks and, finally, mapping tasks to processors. Figure 2 shows a brief parallel design. Detailed description can be found in [11].

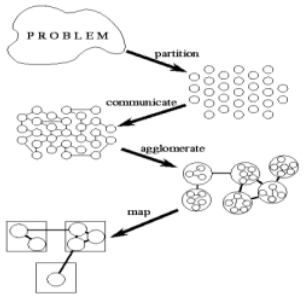

Figure 2 Design methodology for parallel programs [11]

Parallel design, consists of. Partitioning. The computation that is to be performed and the data operated on by this computation are decomposed into small tasks. Communication. The communication required to coordinate task execution is determined, and appropriate communication structures and algorithms are defined. Agglomeration. The task and communication structures defined in the first two stages of a design are evaluated with respect to performance requirements and implementation costs. Mapping. Each task is assigned to a processor in a manner that attempts to satisfy the competing goals of maximizing processor utilization and minimizing communication costs. Partition and communication are the basic principle of desaining parallel algorithm. This paper will discuss these two stages, and other stages will be dicussed as a further research.

III. Result and Discussion

A. Clonal Selection Algorithm Parallelism

Clonal selection is an event in the immune response, whereby an attack of antigens, B-cells as antibody-producing cells, would be multiplied if their receptors match with the antigens' receptors. Cells that do not match the receptor do not participate in the selection. The match calculation is known as affinity maturation. This clonal selection mechanism inspired Leandro and Von Zuben [12] created Clonal Selection Algorithm (CSA) to solve the optimisation problem. This algorithm started with a population generation of B-cells, then an evaluation scheme to check their affinities. If the affinity is good enough (depending on the aim of solution), there are selection mechanisms to get the best individuals. These good individuals (of B-Cells) are then cloned as a clone factor (a scalar value) as defined by programmer. After the cloning process, there is a mutation phase for cloned individuals according to the mutate factor. The cloned and mutated individuals are evaluated and selected to replace the worst individuals in the population. This mechanism is repeated until a stop condition is reached [12].

For the TSP problem, we have to map between CSA parameters and the TSP problem [13] in Table 1 below:

Table 1 Mapping CSA parameters and the TSP problem [13]

Clonal Selection Algorithm step	TSP Problem
Population initialization	Set of randomly generated tours. There are (n-1) possibilities that the tours may be raised. This population is part of the whole tours. The number of tours is generated by the specified population size.
Affinity evaluation	Evaluation of affinity checks each tour that has occurred and finds the cost required to form the tour.
Selection: affinity maturation	Affinity is how close the cost of a tour is to the optimal/best cost. The closer, the higher affinity will be selected.
Cloning and Hypermutation	Cloning is a process to copy a selected tour, so the number of copies depends on the clone factor: β. Cloned/copied tour will be mutated according to hypermutation probability mutate factor: ρ
RandomReplacement	After this mutation, we will have the best tours, which will be replaced by the worst tours in the initial population. The number of worst tours replaced will depend on some random size replacement d.

Clonal Selection Algorithm step	TSP Problem
Stop condition	Clonal selection process will be repeated until a stop condition is obtained. Stopping criteria could be the number of generations, numbers of populations (tours) evaluated, or the best cost found.

To solve the TSP problem, we need a matrix that shows the distances between cities. Here the pseudo code for CSA:

```
Input: matrix of distances
Output: population contains best tours
Process:
Generate random populations of tours
While (not finished)
    Evaluate all tour from the population, check their affinities
    Select best tours from the population
    Clone and mutate the best population
    RandomReplacement
    If ((generation = maxGeneration) || (bestTour found) || (evaluateTours done))
        finished = true
```

CSA consists of one loop that is dependent on each iteration. Since it is dependent, we couldn't do functional decomposition in this loop. The output from a previous iteration will be used in the next iteration.

To find parallelism potentials for CSA, we need to describe each stage: evaluation, selection, clone and mutate, and randomreplacement.

Here we examine the pseudo code for CSA via method evaluation:

```
Input: population of tour, matrix
Output: affinity for each tour
Process:
For each tour in population
    For each element in tour
    Check affinity for each element from matrix
```

The evaluation process involves two loops, which have potential parallelism. Each loop checks the affinity for each element in the matrix that refers to the dataset. This evaluation process indicates the small granularity of jobs, which arises in line with the population size and the complexity of the issue (in a tour length, L). We can assume the complexity of the evaluation process is: population size * the size of the tour = N * L.

Here we examine the pseudo code for CSA, method selection:

```
Input: population of tour
Output: best tour
Process:
    Sort population
    Select tour from population size selectionSize
```

The selection process involves a sequence and selection of the best tour. No potential parallelism can be exploited. Sorting is a fast enough process; when the population data used is not too large. Size complexity of the selection process is according to the sorting algorithm used, we assume NlogN.

Here we examine the pseudo code for CSA, method clone and hypermutation:

```
Input: best population of tour
Output: copy of the best tour that have been mutated already
Process:
Calculate relative normalized affinity
For each tour at best population
    Count its normalizedRelativeScore
    Count number of clone depends on cloneFactor
    For each tour at best population
Count its probality depends on mutateFactor and its normalizedRelativeScore
        For I = 1 to number of clone
            Copy tour
            For each element in tour
            If randomizedNumber <= probabilty then mutate
        Count its affinity
```

Clone and mutate processes are the most complicated process of the algorithm. There are four loops, 1 independent loop and 3 nested loops. The independent loop has the task to calculate the normalised value of each tour. The granularity of this process is not large and no potential parallelism can be exploited in this process.

The three nested loops have tasks as follows: the first loop does iterations of the number of selections, while the second loop does iterations as per the number of clones, and a third loop does iterations of a number of possible mutations. The size and complexity of the clone and mutate processes are = size selection * clone size (clone factor * population size) * (length of elements in a tour, L) = n * (β * N) * L.

Here we examine the pseudo code for CSA, method RandomReplacement:

```
Input: population and clone
Output: best tour
Process:
Sort population
Sort clone population
    Remove the worst tours from population
    Add the best tours from clone population to population
    Get some randomized tours (randomReplacement size)
    Replace the worst tours in population with new randomized tours
```

Randomreplacement is the process of selecting a set worst tour, which is then replaced by the best tour of the cloned mutations. This process conducts sorting of the population and the clone population. There is no repetition of this process. Since the sorting is done by a sorting library and the sorted population is not large, this indicates that this potential parallelism will not be explored further. Size complexity of this process is in accordance with the sorting algorithm used. We assume NlogN and βNlogβN respectively.

B. Parallel Clonal Selection Algorithm

After we describe the parallelism potentials, we then start to parallel design. First step is partition. There are two approaches in partition: domain/data decomposition and functional decomposition. Clonal selection deals with two sets of data: matrix of the dataset and population of tours. Since the algorithm stages need the whole matrix of a dataset to evaluate the affinity, we couldn't do matrix decomposition. Since population of tours are dependent on each other, we could do population partition.

Functional decomposition in CSA deals with stages of evaluation, selection, clone and mutation, and randomreplacement. Since all of these stages are dependent each other, we couldn't do a functional decomposition for each one. There is some parallelism potential in each stage. It is possible to do functional decomposition. Then we have to calculate the granularity for each potential and compare this to whole process. Since we need much generation to evaluate the population, we will need G* (NL+NlogN+nβNL+NlogN+βNlogβN), where G is the number of generation. Granularity is a size of task for each stage. Table 2 shows the granularity for each stage.

We can conclude that functional decomposition in each stage is not recommended since their granularities are too small compared to the whole process. To choose these parallelim potentials, we need some detail exercise for each potential functional decompotition as.a further research. Since we couldn't do functional decomposition for CSA loop, we did population decomposition. Each processing element is assigned to a population partition that will be computed for the CSA stages.

Table 2 Granularity for each stage

Stages	Granularity	Compare to whole process
Evaluation	NL	NL/ G*(NL+NlogN+nβNL+NlogN+βNlogβN)
Selection	NlogN	NlogN/ G*(NL+NlogN+nβNL+NlogN+βNlogβN)
Clone and hypermutation	nβNL	nβNL/ G*(NL+NlogN+nβNL+NlogN+βNlogβN)
RandomReplacement	NlogN+βN logβN	(NlogN+βNlogβN)/ G*(NL+NlogN+nβNL+NlogN+βNlogβN)

The second step to designing parallel computing is communication, a process that transfers data or messages among processes. This communication depends on parallel models: message passing, data parallelism and shared memory [11]. Since the message passing model is probably the most widely used parallel programming model today, we focus on this model.

Message passing means sending and receiving messages to and from tasks. There are multiple tasks, with each task identified by a unique name, and tasks have their own encapsulating local data. These systems are said to implement a single program multiple data (SPMD) programming model because each task executes the same program but operates on different data [14].

In a parallel implementation of CSA, we could design the tasks to execute independently and communicate only in the end of an iteration just to report temporal solutions. We need a master process to control and communicate with other processing elements, called slave processes. This master process takes a role to synchronise a process between slaves' processes and to evaluate temporally the solutions produced by slaves. This design can be seen in Figure 3. In other designs, we could conduct communication among processing elements during execution in order to obtain and update the solutions. There's no master process to synchronise the task, as each processing element communicates with each other by sending and receiving their own temporal solutions. This design can be seen in Figure 4.

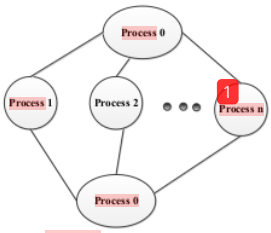

Figure 3 Master-slave communication

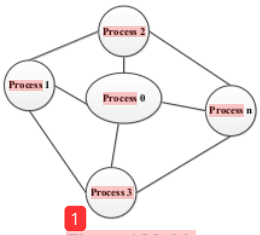

Figure 4 Multi-communication

In the master slave model, the master has task to do:

- Population initialisation
- Send population
- Receive the best population from each slave
- Define the best of the best population and send to slaves again

Slaves do their tasks to:

- Receive population from master
- CSA operator (evaluation selection, clone and hypermutation, replacement)
- Send their own best population to master

In the multi-communication model, each processing element does:

- Population initialisation
  - CSA operator (evaluation selection, clone and hypermutation, replacement)
  - Sending and receiving of each best population to and from other processing elements
  - Calculate the best of the best population
- Repeat until the condition stops

C. Message Passing Model for Parallel CSA

In this section, we introduce a message-passing programming and show how designs developed using the techniques discussed in section B can be adapted for a message-passing execution. In a message-passing parallel programming model, processes do cooperative operations on each process to move data from the address space of one process to another.

We use the Message-Passing Interface (MPI), the de facto message-passing standard for concreteness and MPJExpress, as one of the libraries that implemented MPI. MPI is not a library but rather the specification of what such a library should be [15]. In the MPI programming model, there are two or more processes that communicate by calling library routines to send and receive messages to other processes. Processes can use point-to-point communication operations to send a message from one named process to another. Single Program Multiple Data (SPMD) algorithms, which create just one task per processor, can be implemented with point-to-point or collective communication routines directly.

A master-slave communication model needs to send and receive communication in two ways: using a basic MPI point-to-point communication such as `sending` and `receiving` routines, and or collective communication such as a `broadcasting` routine. In a multicomunnication model, we could use sending-receiving or collective communications such as `allgather`.

D. MPJExpress

The Java Grande forum defined MPI-like API for Java as mpiJava 1.2 API, which is the Java equivalent of the MPI 1.1 specification document. MPJ Express is one type of a reference implementation of the mpiJava 1.2 API. MPJ Express is a message-passing library that can execute parallel Java applications on computed clusters or networks of computers used by application developers [16]. It is like a middleware that supports communication between individual processors of clusters. MPJ Express supported Single Program Multiple Data (SPMD) [17].

There are three ways to configure MPJ Express: multicore configuration, cluster configuration and hybrid configuration. The Multicore Configuration is used to execute MPJ Express user programs on laptops and desktops. In this research, we use multicore configuration [18].

There are three methods we used in our MPI programming model with MPJExpress. Detailed class and methods in the MPI programming model with MPJExpress can be found in [14].

Method: `Send`.

Synopsis:

```
public void Send(java.lang.Object buf,
   int offset, int count,
   Datatype datatype,  int dest,
   int tag)throws MPIException
```

Example in MPJExpress:

```
MPI.COMM_WORLD.Send(objectP, 0, objectP.length,
MPI.OBJECT, dest, tag);
```

Method: `Recv`

Synopsis:

```
public void Recv (java.lang.Object buf,int
offset,int count,Datatype datatype,int source,int
tag)throws MPIException
```

Example in MPJExpress:

```
MPI.COMM_WORLD.Recv(objectP, 0, objectP.length,
MPI.OBJECT, 0, tag);
```

Method: `Allgather`

Synopsis:

```
public void Allgather(java.lang.Object sendbuf,
int sendoffset,int sendcount,Datatype sendtype,
java.lang.Object recvbuf,int recvoffset,
int recvcount,Datatype recvtype)
throws MPIException
```

Example in MPJExpress:

```
MPI.COMM_WORLD.Allgather(sendBuff, 0, unitSize,
MPI.OBJECT, recvBuff, 0, unitSize, MPI.OBJECT);
```

We use object data type to send and receive data, since we need an object to represent a tour in TSP. Rank is an id for each process.

E. Implementations Result

With datasets from TSPLIB, we implemented two model of Clonal Selection Algoritm and result 1) comparison two model, 2) best cost for several dataset and 3) execution time for several dataset. We use a mulicore environment for all expreriments with 4 processing elements, using Toshiba Portege Z935, Prosesor Intel® Core™ i5-3317* CPU@1.70GHz. Memory 4GB, 64-bit Operating System, Netbean Netbean 7.2.1 with Java 1.7.0_13 HotSpot(TM) 64-Bit Server V M 23.7-b01.

Figures below describe all of that three results.

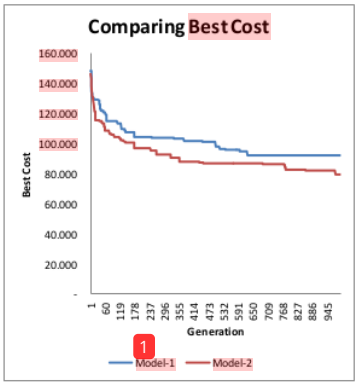

Figure 5 Best cost comparison for kroa100.tsp

We can see that model-2 gained lowes best cost for each generation.

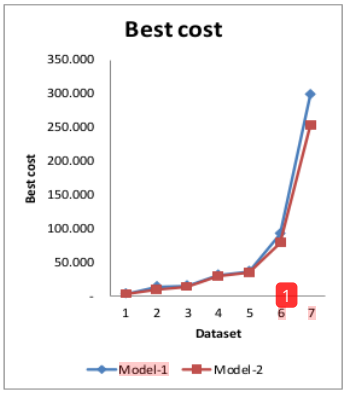

Figure 6 Best cost

Using dataset burma14, berlin52, ulysses22, tsp225, ch150, kroa100, and pr76 from TSPLib, we can see in Figure 6 that model-2 is gained lower best cost for each dataset, but not significant.

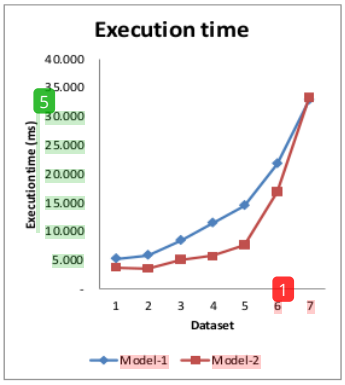

Figure 7 Execution Time

With the same dataset, we can see in Figure 7 that model-2 gained lower execution time, but it gained the same execution time with model-1 in dataset number 7 (pr76.tsp).

IV. Discussion

We described how to implement MJExpress to parallel the clonal selection algorithm, from parallel designing to implementation. The most important part of these steps is parallelism exploitation analysis, especially each of the process granularities. For a population-based optimisation algorithm like this clonal selection algorithms, the best paralleling is achieved by doing their clonal operation at the same time for different population data and then communicating with each other. We have two-way communications, by master-slave communications or by multicommunication. In their implementations, we need a different method library. Using master-slave communication is less simple, or just using point-to-point communication, but it needs synchronisation between population-generation controlled by the master process. Using multicommunication, we could use a global communication, `allgather`, which is more sophisticated but we need to be more careful to define which data has to be sent between processes. From experiment result, we can see that using multicommunication model, we gained lower best cost dan lower execution time.

V. Conclusion And Future Work

To parallelise population-based algorithms, we need to exploit and define their granularity for each stage. After that we do data or functional partition, if necessary, and use the communication model. Using a library for the message-passing model, we should define appropriate methods to implement process communication.

MPJExpress, as one of the implementations of the message-passing interface specification, has several communication methods that fit in with the SPMD model.

Two kinds of communication between processes can be executed for several datasets from the TSP problem, and then we can compare their performances using execution time and the best cost achieved. We conclude that multicommunication model will have better performance than master slave model.

As a future research, we will do some elaboration about choosing functional decomposition with their each granularity, several number of processing elements, and larger datasets. We also can implement using two kinds of MPJExpress architecture, multicore and multicomputer then compare the results.

References

[1] Donald Davendra, *Traveling Salesman Problem, Theory and Applications*, Donald Davendra, Ed. Rijeka, Croatia: InTech, 2010.

[2] Enrique Alba, *Parallel Metaheuristic: A New Class of Algorithms.*: A John Wiley & Sons, Inc., 2005.

[3] Teodor Gabriel Crainic, Tatjana Davidović, and Dušan Ramljak, "Designing Parallel Meta-Heuristic Methods Designing Parallel Meta-Heuristic Methods,".

[4] Jonathan Timmis and Leandro Nunes Castro, *Artificial Immune Systems: A New Computational Approach*. London: Springer Verlag, 2002.

[5] Kulturel-Konak S Ulutas BH, "A Review of Clonal Selection Algorithm and Its Applications," *Artificial Intelligence Review*, 2011.

[6] Alsharhan S, J.R. Al-Enezi. Abbod MF, "Artificial Immune Systems – Models , Algorithms and Applications," *International Journal of Research and Reviews in Applied Science (IJRRAS)*, pp. 118-131, May 2010.

[7] Gaber J Bakhouya M, "An Immune Inspired-based Optimization Algorithm : Application to the Traveling Salesman Problem," *AMO - Advanced Modeling and Optimization*, vol. 9, no. 1, pp. 105-116., 2007.

[8] Jacek Dabrowski and Marek Kubale, "Computer Experiments with a Parallel Clonal Selection Algorithm for the Graph," in *IEEE International Symposium on Parallel and Distributed Processing*, Miami, FL, 2008, pp. 1-6.

[9] Zhu Hongbing, Chen Sicheng, and Wu Jianguo, "Paralleling Clonal Selection Algorithm with OpenMP," in *3rd International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Shenyang, 1-3 Nov. 2010, pp. 463 - 466.

[10] Purbasari, Ayi, "Using parallel clonal selection algorithm to solve multi travelling salesperson problem", in *6th International Workshop on Computer Science and Engineering, WCSE*, Tokyo, 17-19 June 2016

[11] Ian Foster. (1995) Designing and Building Parallel Programs. [Online]. http://www.mcs.anl.gov/~itf/dbpp/

[12] Leandro N. de Castro and Fernando J. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle," *IEEE Transactions On Evolutionary Computation*, vol. 6, no. 3, pp. 239-251, June 2002.

[13] Purbasari, Ayi, "Data Partition and Communication on Parallel Heuristik Model Based on Clonal Selection Algorithm", TELKOMNIKA, vol Vol13, No1, 2015, pp 193 - 201

[14] Blaise Barney. (2012) Introduction to Parallel Computing. [Online]. https://computing.llnl.gov/tutorials/parallel_comp/#Designing

[15] Argonne National Lab Mathematics and Computer Science. The Message Passing Interface (MPI). [Online]. http://www.mcs.anl.gov/research/projects/mpi/

[16] Mark Baker and Bryan Carpenter, "MPJ: A New Look at MPI for Java," in *Poster Paper in All Hands Meeting (AHM)*, 2005.

[17] Mark Baker, Bryan Carpenter, and Aamir Shafi, "MPJ Express: towards thread safe Java HPC," in *IEEE International Conference on Cluster Computing*, 2006.

[18] Aamir Shafi, Jawad Manzoor, Kamran Hameed, Bryan Carpenter, and Mark Baker, "Multicore-enabling the MPJ Express messaging library," in *The 8th International Conference on the Principles and Practice of Programming in Java - ACM*, 2010.

clonal Selection Algorithm Parallellization With MPJExress

ORIGINALITY REPORT

97%	44%	97%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ayi Purbasari. "Clonal Selection Algorithm parallelization with MPJExpress", 2016 8th Computer Science and Electronic Engineering (CEEC), 2016 Publication	74%
2	toc.proceedings.com Internet Source	21%
3	"[Copyright notice]", 2021 14th International Conference on Human System Interaction (HSI), 2021 Publication	2%
4	"Cover page", 2016 8th Computer Science and Electronic Engineering (CEEC), 2016 Publication	1%
5	riuma.uma.es Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		