(IJNCAA)

ISSN 2220-9085(ONLINE)
ISSN 2412-3587(PRINT)

INTERNATIONAL JOURNAL OF

New Computer Architectures and their Applications

Volume 7, Issue 3
2017

SDIWC

www.sdiwc.net

International Journal of
NEW COMPUTER ARCHITECTURES AND THEIR APPLICATIONS

Overview

The SDIWC International Journal of New Computer Architectures and Their Applications (IJNCAA) is a refereed online journal designed to address the following topics: new computer architectures, digital resources, and mobile devices, including cell phones. In our opinion, cell phones in their current state are really computers, and the gap between these devices and the capabilities of the computers will soon disappear. Original unpublished manuscripts are solicited in the areas such as computer architectures, parallel and distributed systems, microprocessors and microsystems, storage management, communications management, reliability, and VLSI.

One of the most important aims of this journal is to increase the usage and impact of knowledge as well as increasing the visibility and ease of use of scientific materials, IJNCAA does NOT CHARGE authors for any publication fee for online publishing of their materials in the journal and does NOT CHARGE readers or their institutions for accessing the published materials.

Publisher

The Society of Digital Information and Wireless Communications
20/F, Tower 5, China Hong Kong City, 33 Canton Road, Tsim Sha Tsui, Kowloon, Hong Kong

Further Information

Website: http://sdiwc.net/ijncaa, Email: ijncaa@sdiwc.net,
Tel.: (202)-657-4603 - Inside USA; 001(202)-657-4603 - Outside USA.

The issue date is September 2017.

IJNCAA

ISSN 2220-9085 (Online)
ISSN 2412-3587 (Print)

Volume 7, Issue No. 3 *2017*

CONTENTS

ORIGINAL ARTICLES

International Journal of New Computer Architectures and their Applications (IJNCAA) 7(3): 68-76
The Society of Digital Information and Wireless Communications, 2017 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Speeding Up Canny Edge Detection Using Shared Memory Processing

Njood S. Alassmi and Soha S. Zaghloul, PhD.
College of Computer & Information Sciences, Department of Computer Science, King Saud University, Riyadh
435920199@student.ksu.edu.sa, smekki@ksu.edu.sa

ABSTRACT

Image processing is a computational operation that requires many CPU cycles for simple image transformation. It takes every pixel of an image to perform a transformation to a new image. The image can be divided into smaller chunks, with same transformation operations being implemented on each. Thus, image processing is a good candidate for running on a parallel processor to improve the speed of computation when there are multiple images to be processed. In fact, this research focuses on Canny edge detection as a case study of probing parallelism. This work presents the design and implementation of sequential and parallel edge detection algorithms that are capable of producing high-quality results and performing at high speed. Therefore, this research aims to improve the Canny edge detection algorithm in terms of speed and scalability with different sizes of images. The algorithm is implemented using Java language with the Parallel Java library. In the first phase of the project, the Canny is implemented on a shared-memory processor. It is found that there is a valuable gained speedup with respect to the sequential version. In addition, it is found that more parallelism is explored in larger image sizes with Canny edge detector.

KEYWORDS

Canny Edge Detection, Shared-Memory, Sequential Implementation, Parallel Implementation, Parallel Java Library.

1 INTRODUCTION

The growth of the world's technological industry has an important effect on our lives; we are witnessing a growth of computer power combined with a development of the digital camera abilities. Currently, it is important for a large number of users to assess the quality of acquired images. Image edge detection is one of the operations implemented with the purpose of an image refinement. The edge detection process is significant in image analysis including texture feature extraction, image segmentation and shape feature extraction. Recently, it becomes the most effective field of the digital image processing. There are several approaches available to detect the edges of an image such as Prewitt operator, Canny operator, Roberts operator and Sobel operator. However, to create a high-quality edge detection method, more complexity is added to get high-quality that is able to be performed at high speed.

In order to achieve high performance computing (i.e. reducing computing elapsed time), parallel processing is widely used in multimedia computing, signal processing, scientific computing, engineering, general purpose application, industry, computer systems, statistical applications, and simulation [1]. The concept of parallel image processing is adapted for the purpose of fast image processing. The basic concept behind parallel computing is the distribution of the workload between the individual processors that are working together to perform computations. Various image processing techniques can be easily implemented in a parallel fashion [2]. In order to apply shared memory, we need parallel programming languages for implementation instead of the traditional sequential ones. The aim of this research is to decrease the execution time of an existing image edge detection algorithm by comparing the execution times of both the serial and parallel programs. The objectives may be listed in the following points:

- Increase the gained speedup by implementing the algorithm on a parallel environment.
- Explore the efficiency and scalability of the parallel algorithm.

- Improve the Canny edge detection algorithm in terms of speed.

This research focuses on Canny edge detection as a case study to probing parallelism. The Canny operator represents a multi-stage process. Firstly, the Gaussian filter is applied to smooth the entire image. Secondly, the gradient magnitude for the image and the upper and lower threshold values are determined. The tracking process uses hysteresis controlled by two thresholds: T1 and T2. This hysteresis helps to ensure that noisy edges are not broken up into multiple edges. Edge detection is an important area in many applications in the field of image processing. It is a foundational step for high-level information acquisition because image edges contain relevant information. Currently, it plays a critical role in many fields, such as mechanics, the military and medical diagnosis.

2 REVIEW OF RELATED STUDIES

This research focuses on Canny edge detection as a case study to implement the parallelism concept. In [3], the author presented the Canny edge detection operator that used a multi-stage algorithm to detect a broad range of edges in images. The aim of Canny is to develop an optimal algorithm focusing on the following criteria: Firstly, detection (the probability of detecting real edge points) should be increased, while the probability of detecting non-edge points should be decreased. Secondly, localization defined as the detected edges should be as similar as possible to the real edges. Conversely in [4], the authors stated that Canny edge detection algorithm should have a good rate of detection with a minimum number of false edges, good localization for which proximity of the real edges and the detected edges. Finally, minimal response means one edge should be detected only once. In order to implement the Canny edge detection algorithm, a series of steps must be followed. The first is to filter out any noise in the original image before attempting to detect any edges. This can be done by applying the Gaussian filter that can be computed using a simple mask. After smoothing the image and removing the noise, the next step is to find the edge strength by measuring the gradient of the image. Subsequently, the direction of the edge should be computed using the gradient in the x and y directions. After the edge directions are known, non-maximum suppression should be applied, which is used to trace along the edge in the edge direction and suppress any pixel value. This results in a thin line in the output image. The final stage is edge tracking by hysteresis; the final edges are determined by suppressing all edges that are not associated with the definite edge [5].

In [6], the authors tried to use the Canny operator with high quality in a short time. They found that the execution time is mostly 0.2184 second for input images with a size of 512*512 pixels. In [5], the authors discussed the advantages and disadvantages of edge detection techniques. The Sobel operator provided simplicity, as well as the detection of edges and their orientations. On the other hand, its disadvantages include being sensitive to noise and inaccurate. In contrast, the Canny operator using a Gaussian filter removes any noise in an image, yielding a better detection of edges. Y. Kong et al. [7] implemented the Canny edge detection method and found that almost 90% of the edges are correctly detected. However, disadvantages include its intensive and complex computations, ensuring that it is time consuming. The overall time complexity is found to be O (m*n log m*n), where m and n are the width and length of the image in terms of pixels.

Image edge detection algorithms are implemented in various parallel environments, including GPUs, SMPs, and MPPs. In [8] and [9], the authors proved the efficiency of the GPU in conducting in parallel the millions of pixel calculations involved in image processing. In [8], the authors examined the edge detection in both sequential and parallel algorithms in order to measure the speedup. The results indicate that the parallel implementation is about 262 times faster than the sequential implementation. In [9], the authors proposed a parallel Canny algorithm for the embedded CPU and GPU heterogeneous. The results

showed that the proposed parallel Canny algorithm achieved nearly 50 times speedup on the embedded systems. They used differently sized images for the tests. For 512×512 images, the runtime of a Canny algorithm in sequential implementation is found to be over 1,5898 milliseconds. Otherwise, Canny improved the runtime to a level of 3,310 milliseconds by using embedded CPUs and GPUs.

3 SEQUENTIAL EDGE DETECTION ALGORITHM

The applied sequential edge detector is Canny's since it is known to provide the optimal edge detector [10]. Canny's detector applies many steps before reaching the final result. The input image is read into an array of size m×n where m and n are the width and length of the image respectively in terms of number of pixels. The algorithm starts by storing the 2-D image in a 1-D array to simplify processing in later phases. Then, the algorithm visits every pixel of the image and makes the modifications as detailed below. The algorithm goes through a series of steps [11]:

1. **Noise Removal:** The first step is to filter any noise in the original image before attempting to detect edges. This is done using Gaussian filter computed by a simple mask and exclusively used in the Canny algorithm.
2. **Gradients Calculation:** After eliminating the noise and smoothing the image, gradients of the image are taken for each edge, gradient magnitude (M) represented the edge intensity.
3. **Thresholding:** Two thresholds are used to detect image edges. Two values, upper and lower, are specified. This allows for greater flexibility than using a single threshold approach.
4. **Edges Classification:** The thresholding is used to detect the final edges in an image by using the two thresholding values of v_low and v_high. In this step, the pixels are classified into three categories in terms of the value of the magnitude M. If a pixel gradient is higher than the high threshold (M > v_high), the pixel is accepted as a strong edge. If (v_low ≤ M <v_high), the pixel is classified as a weak edge and waits for the judgement of the edge points. If (M <v_low), the pixel is rejected if classified as a non-edge. The diagram in Fig. 1 shows the main steps of the approach for Canny edge detection.

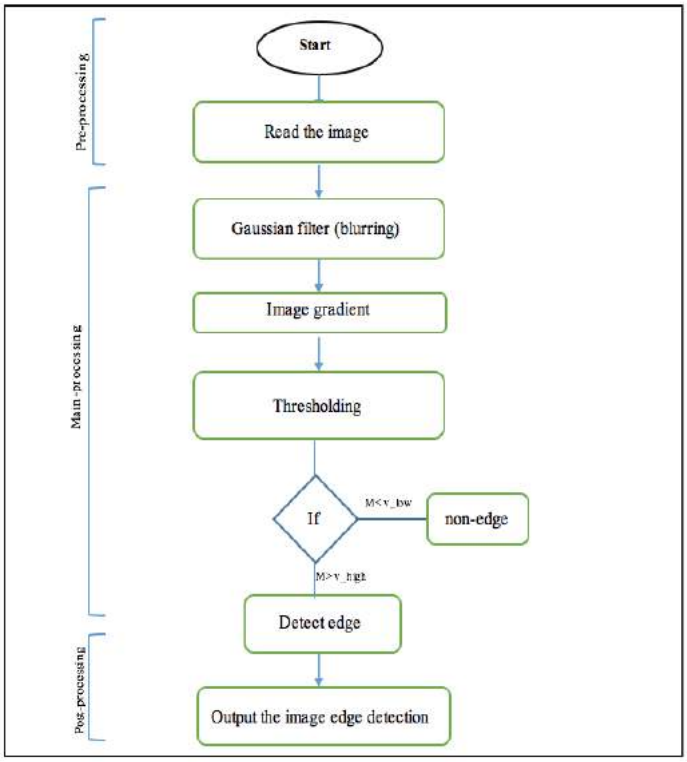


Figure 1. The flow chart of the sequential version for Canny Edge Detection steps

4 PARALLEL EDGE DETECTION ALGORITHM USING SMP

SMP is a symmetric multiprocessor; each processor has its own cache. As the name indicates, the processors communicate with each other through a shared memory. A process is divided into threads and each thread runs on a core. Consequently, the memory can be together accessed directly by multiple processors. In this situation, communications among processors are carried out by accessing the same memory space from different processors. This scheme makes it easier for a developer to write in parallel program, by making at least the data sharing much more appropriate. An advantage of SMP is that data sharing between tasks is fast and uniform due to the closeness of memory to CPUs. On the other hand, the main disadvantage is the lack of scalability between memory and CPUs.

Parallelism is applied by looking for independency between data, especially in loop iterations. Canny edge detection algorithm is a good candidate for parallelism. The reason behind that is the available independent processing required on image pixels. On the other hand, the available hardware defines the applicable parallelism level. For example, processing an image of resolution 512×512 implies that the number of operations in a single step of Canny algorithm is 262,144. When run on dual-core computer, these computations are divided into two chunks; each of which handles 131,072 operations running simultaneously. In this phase of the research, a parallel algorithm is designed for an SMP by dividing the tasks of pixels' modifications by the number of cores. Each chunk is assigned to a core, and operations run simultaneously. In other words, each core is responsible for processing its own segment of array at the same time. Once the processing is done, the new data array is produced. The latter is then written onto a new image file, generating the image edges as detected by Canny parallel algorithm. The following pseudo-code explains the parallel Shared-memory for Canny edge detection steps using parallel java PJ2:

Algorithm 2: The parallel Shared-memory for Canny Edge Detection

```
start
   1- Read Input Image I;
   2- Obtain the Image Size S;
   3- Set the Gaussian_valu, LowThreshold_valu,
      HighThreshold_valu;
StartTime = System.currentTimeMillis();
   4- new Parallel Team ().execute (new ParallelRegion()
       {
         public void run () throws Exception
            {
           execute (0, S - 1, new IntegerForLoop ()
               {
                 public void run (int first, int last)
                    {
             for (int x = first; x <= last; x++)
            {
                  //Apply Gaussian_filter g (x, y);
             }
               }
                 public void finish ()
            {
            //Performing to get the final result after the
   Parallel Job
             }
         });
   }
}); //end Parallel Team.
   5- Apply the Parallel Team on the Gradient magnitude;
   6- IF (M > v_high)
       Accepted as a strong edge;
      else
      The pixel is rejected;
EndTime = System.currentTimeMillis();
   7- Output the image edge detection and Time taken.
end
```

5 EXPERIMENTAL RESULTS

5.1 Software and Hardware Requirements

In this research, the Canny image edge detector is implemented in both sequential and parallel approach. The parallel version is applied in a SMP environment. Java programming language is used in coding with the support of Parallel Java library to manage threads under Linux Ubuntu 16 operating system.

On the other hand, a Toshiba laptop featured with an Intel quad-core i5 processor with 2.4 GHz speed is used in the current phase of the research. The processor does not support hyper-threading; therefore, the maximum number of experimented threads is four. The memory capacity is 8 GBytes.

5.2 Performance Measurements

This research aims to increase the speedup, the efficiency and the scalability of the sequential code by exploring its embedded parallelism.

The performance of the parallel algorithm is measured by comparing it to the sequential algorithm. Three main metrics are used to measure such improvement; namely, the speedup, the efficiency and the scalability. These are detailed in the following sections.

- Speed up

In order to differentiate between the running times of the sequential and parallel algorithms, the following notations are used: T_{seq} denotes the running time of the sequential program, and T_{par} denotes the running time of the parallel program. The speedup (S) is defined as the ratio of the execution time of the serial implementation over the execution time of the parallel implementation. Let T (N, 1) be the time required for the best serial algorithm to solve problem of size N on one processor and T (N, K) be the time for a given parallel algorithm to solve the same problem of the same size N on K processors. Thus, speedup is defined as [12]:

$$S(N, K) = \frac{T_{seq}(N, 1)}{T_{par}(N, K)} \quad (1)$$

-Efficiency

Efficiency (E) is a metric that how close a program is to ideal speedup. The efficiency of a real program is less than **1**. A program is close to the ideal status

as its efficiency is closer to 1. Efficiency is therefore measured using the following formula [13]:

$$E(N, K) = \frac{\text{Actual Speedup}}{\text{Ideal Speedup}} = \frac{\text{Speedup } (N,K)}{K} \quad (2)$$

-Scalability

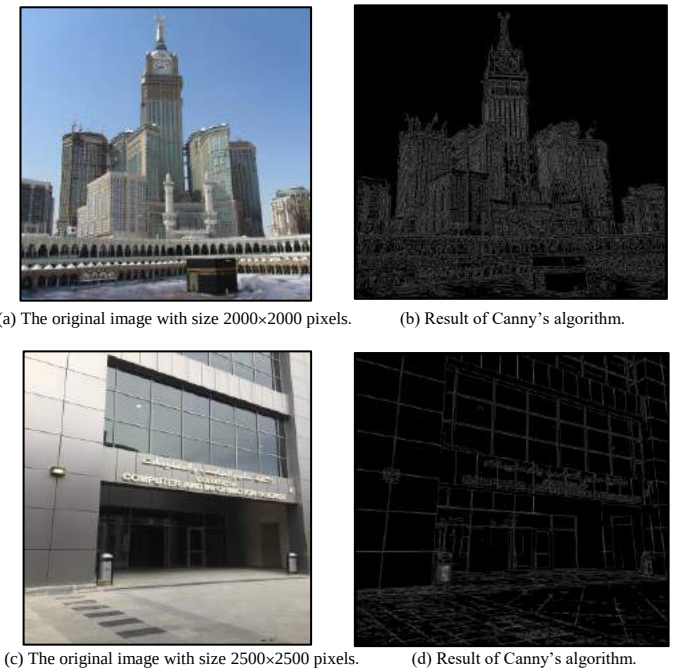
A program's size-up is the size of the parallel version running on K processors relative to the size of the sequential version running on one processor for a given running time T [13]:

$$\text{Size-up } (T, K) = \frac{N \text{ par } (T, K)}{N \text{ seq } (T, 1)} \quad (3)$$

6 RESULTS

6.1 Sequential Code

This research uses three general images of different sizes; namely, 1024x1024, 2000×2000 and 2500×2500 pixels. The initial image sizes are modified to the previously mentioned sizes using Adobe Photoshop software. Fig. 3 (a, c) show the original images; whereas Fig. 3 (b, d) expose them after applying the algorithm. It can be detected that the edges are clear in the output images. Thus, it can be concluded that the Canny algorithm is suitable for these samples of colored images.



(a) The original image with size 2000×2000 pixels. (b) Result of Canny's algorithm.

(c) The original image with size 2500×2500 pixels. (d) Result of Canny's algorithm.

Figure 3. The sequential edge detection results

Experiments are applied on three image sizes to measure the running time. The results are shown in (Table 1) in the Appendix. Multiple image sizes are used to explore the relation between the image sizes and the corresponding execution times. It is found that the execution time is directly proportional with the image size. Such result is expected since the problem size increases.

Fig. 4 shows the relation between the image size and the execution time as measured by the sequential program. The image size is in pixels whereas the execution time is measured in seconds. Expectedly, the execution time is directly proportional with the image size. Precisely, it is found that for the image size 1024×1024 pixels, it took about 0.569 seconds. While for the image size 2000×2000 pixels, the execution time took 0.868 seconds with an increase in time of 54.3% as compared to the 1024x1024 image. On the other hand, it took 1.39 seconds to detect the edges of an image of size 2500x2500 pixels. More precisely, the execution time increased by 58.3% as compared to the second image. Obviously, the execution time increases exponentially with the increase of image size.

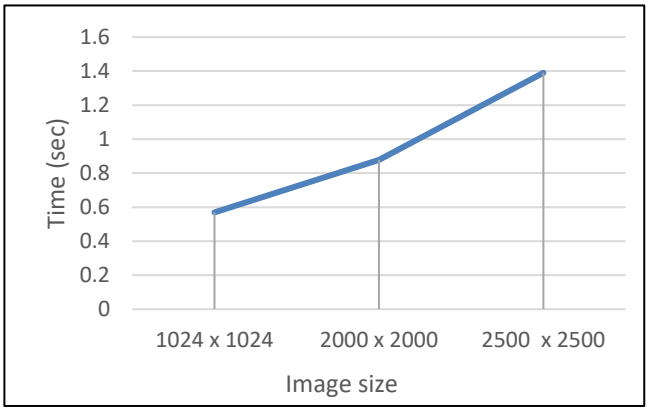


Figure 4. The runtime for sequential edge detection

6.2 Parallel Code

In order to get accurate results, both sequential and SMP versions are tested on the same computer to exclude the hardware impact on the resulting execution times. In addition, the basic algorithm is the same in both versions. Moreover, the SMP code is tested with the same images previously used in the

experiments of the sequential code. Fig. 5 shows the resulting detected edges using the sequential and SMP codes in (a) and (b) respectively.

(a) (b)

Figure 5. The edge detection results by using SMP

It is noticed that there is a difference in the produced edges for each image. More precisely, the parallel implementation is less sensitive to weak edges as compared to the sequential code. Therefore, the images resulting from the parallel implementation is closer to real edges than its sequential counterpart.

Table 2 in the Appendix details the experimental results on a shared memory processor for the three sizes of the image with various numbers of cores. The parallel code is run for each image ten times for two cores, then ten times for the three cores, and ten times on four cores. For each batch, the minimum execution time is taken. The execution times of the sequential version are added in the last row for convenience.

It is noticed that the execution time of the SMP code is reduced for both two and three cores as compared to the sequential version. Also, for each image size, the recorded execution time on three cores is less than its corresponding one on two cores. However, the four-core results are not always the minimum as may be expected. This may be explained by the imposed overhead of threads scheduling which increases with the increasing number of cores.

More precisely, the execution times for the image size 1024×1024, 2000x2000 and 2500x2500 pixels are 420, 687 and 1126 respectively when run on two cores. Compared to the sequential algorithm, a reduction of 35.5%, 28.2% and 18.99%, this is decrease execution time at a rate of 35.48% for sequential execution. While the second image with the size 2000×2000 pixels, the execution time took 678 milliseconds at a rate of decrease of 28.17% from the execution time in the sequential. The last image with size 2500×2500 pixels, the execution time took 1126 milliseconds. The difference in execution time from the previous image is decreased by 23.45%. Also, the implementation of 3-cores has less execution time than the previous execution.

Fig. 6 depicts the results recorded in Table 2 in the Appendix. Obviously, the execution time for all image sizes is the largest in the sequential code, and the least when executed on three or four cores, since they almost have the same figures. These results are in fact expected.

Figure 6. Comparison between Shared-memory(SMP) and sequential execution

Table 3 in the Appendix displays the gained speedup and efficiency of the SMP Canny edge detection algorithms for the three images' sizes. The table proves experimentally that the execution time decreases with the increase of number of cores.

In Fig. 7, the speedup slightly increases for the 1024×1024 image when run on four cores as compared to the 2-cores and 3-cores. The increase is higher as the image size increases. This demonstrates that more parallelism is explored in larger image sizes with Canny edge detector. Although the speedup increases with the increase of cores, but the efficiency decreases for all image sizes as shown in Fig. 8. This is also expected since the ceiling of the ideal speedup increases with larger number of cores. On the other hand, the actual speedup does not rise up equally. Therefore, the efficiency decreases with the increase of the number of processors.

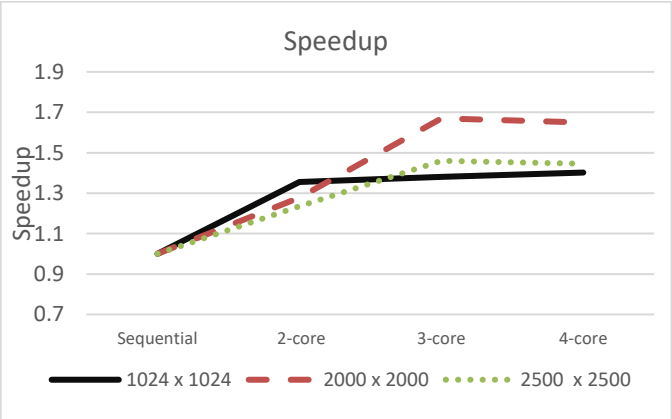


Figure 7. The speedup of the SMP Canny edge detection algorithms

Figure 8. The efficiency of the SMP Canny edge detection algorithms

Fig. 7 shows that the gained speedup of using two threads is around 1.4 for the smallest image size. However, the gained speedup decreases with the increase of image size. On the other hand, the results for three-core and four-core architectures are almost the same. It is also noted that the gained speedup is around 1.6 at its maximum for an image size of 2000x2000; it then drastically degrades for the largest image size. This might be explained by the limited capabilities of the used laptop. This explanation is proved in Fig. 8, where the efficiency is at its best for two threads, and at its minimum for four threads.

Fig. 9 displays the scalability of the SMP Canny edge detection algorithms for the three images' sizes. The scalability factor, as previously explained in Section 5.2, or the sizeup efficiency, is calculated as follows:

$$\text{Scalability factor (2-cores)} = \frac{N2}{N1} = \frac{3000}{2500} = 1.2$$

$$\text{Scalability factor (3-cores)} = \frac{N3}{N1} = \frac{3000}{2500} = 1.2$$

$$\text{Scalability factor (4-cores)} = \frac{N4}{N1} = \frac{3000}{2500} = 1.2$$

Where N_i represents the largest problem size that can be handled when run using *i* cores. As noticed, the largest problem size is 2500x2500 and 3000x3000 for sequential and multi-cores respectively. Although the largest problem size is identical for multi-cores, but the execution time decreases for this size when run on three and four cores. As previously

mentioned, the main disadvantage for SMP is the lack of scalability of memory and the number of cores/threads. Such limitations are overcome when using SANAM supercomputer as planned for the next project phase.

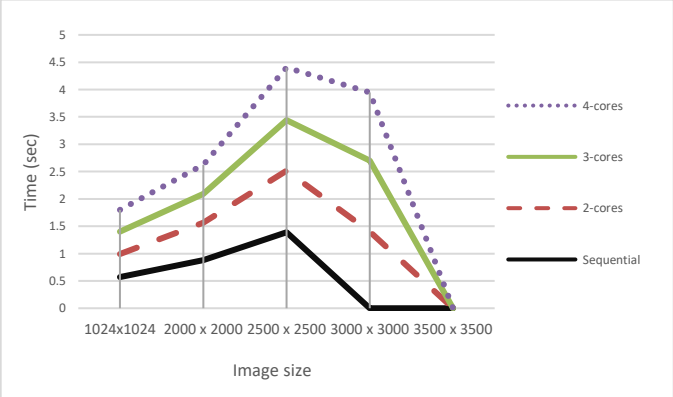


Figure 9. Results of parallel Scalability (Sizeup)

7 CONCLUSION & FUTURE WORK

This project implements image detection twice: the first time using a sequential code, and the second time using a parallel code for a shared memory processor using 2, 3 and 4 cores. It is proved that the Canny edge detection technique performs at a higher speed in parallel environment than sequential. The performance is measured based on the speedup, efficiency and scalability. It is found the average gained speedup is equal to 1.3, 1.4, and 1.50 for two, three, and four cores respectively. Therefore, it can be concluded that the gained speedup increases with a larger number of cores. On the other hand, the experiments found that the average efficiency is 0.64, 0.50, and 0.37 for two, three, and four cores respectively. In conclusion, the efficiency decreases with the increase of the number of cores since the actual speedup does not increase at the same rate as the linear speedup.

In addition, it is found that more parallelism is explored in larger image sizes with Canny edge detector. Also, the experiments show that the sizeup efficiency is 1.2 for all number of cores used in the experiment. In fact, the main disadvantage of SMP is the lack of scalability of memory and number of cores.

In the future, a third version of Canny edge detector is to be implemented on SANAM supercomputer clusters. The objective is to explore the three previously mentioned performance metrics on a cluster environment that contains thousands of processors.

REFERENCES

1. H. Khalilieh, “Performance Evaluation Of Message Passing Vs. Multithreading Parallel Programming Paradigms On Multi-Core Systems,” International Journal of New Computer Architectures and their Applications, vol. 4, no. 4, pp. 108–116, 2014.
2. Y.-M. Kang, “Gpu-Based Object Identification In Large-Scale Images For Real-Time Radar Signal Analysis,” International Journal of New Computer Architectures and their Applications, vol. 6, no. 4, pp. 140–149, 2016.
3. Y. Cheng, “An improved Canny Edge Detection Algorithm,” in Recent Advances in Computer Science and Information Engineering, vol. 126, Springer, pp. 551–558, 2012.
4. Shokhan M. H.,’’ An Efficient Approach for Improving Canny Edge Detection Algorithm,’’ International Journal of Advances in Engineering & Technology, Vol. 7, Issue 1, pp. 59-65, 2014
5. M. Raman R., and H. Aggarwal. "Study and comparison of various image edge detection techniques," International journal of image processing (IJIP) Vol.3 no.1, 2009.
6. M. R. Vahidi, M. M. RiahiKashani and A. Bagheri, “An efficient gradient based algorithm for improving performance of image edge detection,” International Journal of Computer Applications, 103(4), 2014.
7. A.-A.-N., Y. Kong, and M. N. Hasan, “Performance analysis of Canny's edge detection method for modified threshold algorithms,” 2015 International Conference on Electrical & Electronic Engineering (ICEEE), 2015.
8. A. Jain, A. Namdev and M. Chawla, "Parallel edge detection by SOBEL algorithm using CUDA C," 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-6, 2016.
9. D. Yuefan. “Applied Parallel Computing,” Singapore, US: Imperial College Press, 2012.
10. A. Muntasa, “Hybrid Method Based Retinal Optic Disc Detection,” International Journal of New Computer Architectures and their Applications, vol. 5, no. 3, pp. 102–106, 2015.
11. G. M. H. Amer and A. M. Abushaala, "Edge detection methods," 2015 2nd World Symposium on Web Applications and Networking (WSWAN), Sousse, pp. 1-7, 2015.
12. E. Guerrout*, R. Mahiou and S. Ait-Aoudia, “Medical Image Segmentation on a Cluster of PCs using Markov

Random Fields," International Journal of New Computer Architectures and their Applications, pp. 35-44, 2013.

13. A. Kaminsky, "BIG CPU, BIG DATA: Solving the World's Toughest Computational Problems with Parallel Computing" 2nd ed, Boston, MA: Course Technology, Cengage Learning, 2015.

APPENDIX

Table 1. The execution time of the sequential edge detection.

Image size / #	1024 × 1024	2000×2000	2500 × 2500
1	581	924	1454
2	629	878	1461
3	599	883	1433
4	577	884	1422
5	593	910	1390
6	583	937	1412
7	585	880	1478
8	598	891	1440
9	567	882	1400
10	589	905	1412
Minimum	**569ms**	**878ms**	**1390ms**

Table 2. The execution time for Shared memory(SMP).

Image size / #	1024×1024			2000×2000			2500×2500		
	2-cores	3-cores	4-cores	2-cores	3-cores	4-cores	2-cores	3-cores	4-cores
1	446	416	439	716	566	571	1165	1062	981
2	425	432	438	719	551	544	1194	1099	992
3	441	433	433	688	557	542	1170	1052	983
4	451	419	413	665	526	567	1176	1017	974
5	444	418	419	687	558	553	1126	966	961
6	424	412	429	684	547	543	1154	1002	972
7	424	423	410	735	568	551	1188	977	983
8	420	424	422	671	568	543	1150	1025	974
9	437	432	406	681	586	535	1153	952	991
10	457	426	420	696	584	550	1144	977	979
Minimum	**420ms**	**412ms**	**406ms**	**687ms**	**526ms**	**535ms**	**1126ms**	**952ms**	**961ms**
Sequential	**569ms**			**878ms**			**1390ms**		

Table 3. Results of parallel execution time, speedup and efficiency.

Image size	Number of cores	Execution time(milliseconds)	Speedup	Efficiency
1024x1024 pixels	1 (Sequential)	569	1	1
	2	420	1.355	0.677
	3	412	1.381	0.463
	4	406	1.402	0.351
2000x2000 pixels	1 (Sequential)	878	1	1
	2	687	1.278	0.639
	3	526	1.670	0.556
	4	535	1.641	0.410
2500x2500 pixels	1 (Sequential)	1390	1	1
	2	1126	1.235	0.617
	3	952	1.460	0.486
	4	961	1.446	0.362

International Journal of New Computer Architectures and their Applications (IJNCAA) 7(3): 77-88
The Society of Digital Information and Wireless Communications, 2017 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Drone Route Planning for Military Image Acquisition Using Parallel Simulated Annealing

Eman Alsafi and Soha S. Zaghloul, PhD
King Saud University
435204448@student..ksu.edu.sa
smekki@ksu.edu.sa

ABSTRACT

Recently, drones became one of the most interesting technologies used in diverse applications. One of the major challenges associated with them is to set a short route plan in the minimum period of time. This is an important inquiry in many applications, especially in military ones. On the other hand, the simulated annealing algorithm is one of the metaheuristics selected to generate a feasible solution to solve such problem. However, the algorithm requires long execution time. Therefore, the simulated annealing algorithm is parallelized in order to find a suitable solution of the problem in less time. Parallel programming divides the problem into smaller independent tasks, and then executes the sub-tasks simultaneously. Three performance metrics are used to measure the effectiveness of the parallel program; namely, the gained speedup, the efficiency and the scalability. Experiments are conducted to compare the performance of the sequential and parallel version of simulated annealing algorithm. In this paper, the algorithm is implemented on a shared-memory processor environment. The results reported in this research show that the performance of the parallel simulated annealing on shared memory processor is higher than its sequential counterpart in almost all aspects. In addition, the given final distance is less than the sequential simulated annealing.

KEYWORDS

Parallel programming, shared memory, simulated annealing, parallel java.

1 INTRODUCTION

Within the past decade, drones became one of the most interesting technologies. Drones are remotely controlled pilotless aircrafts. They are usually associated with sensors and devices such as cameras, computing units, communication tools, and others [1, 2]. Drones are utilized in diverse military and civil applications. Examples include, but are not limited to, aerial surveillance, image acquisition, remote sensing, and other aspects of scientific research [2]. Drones gained such popularity because they have several advantages over traditional aircrafts; the most important of which is saving human lives in dangerous areas and completing missions quickly with minimum cost [1, 3]. On the other hand, drones have a few limitations; such as limited flight time and energy. Therefore, one of the main challenges when dealing with drones is to find an effective route plan in a suitable amount of time [4].

As drones usually follow preloaded instructions without human intervention, the route plan may be generated either online during the flight, or offline before taking off. Moreover, drones route planning becomes more challenging when there are several geographical locations to be visited that are dispersed apart; these are called waypoints [2]. The aim of this research is to find a route plan that allows drones to acquire images from predefined waypoints in the least possible amount of time. In addition, each waypoint is to be visited exactly once. Obviously, this is analogous to the well-known Travelling Salesman Problem (TSP). Finding a near-optimum route plan is necessary to minimize the drone's power consumption during the flight in order to cover the largest possible geographical area; and therefore, visit the largest number of targeted waypoints.

However, solving TSP problem using a brute-force approach requires a significant amount of time to try every possible solution [2]. This approach is not suitable for the problem in-hands, since time and secrecy are both important factors in military applications. Therefore, a metaheuristics algorithm, the simulated annealing (SA) algorithm is used. In fact, SA is capable of finding an acceptable local optimum route plan

[5]. Although SA is used to solve several complex problems, but it requires significant processing time to find a suitable solution [6]. In order to overcome such drawback, parallel computing is expected to positively contribute in the solution of this problem. Parallelism may minimize the execution time to fulfill the requirements of the military mission. In addition, it may increase the chance to provide a better-quality plan. However, one of the big challenges associated with parallel SA is that the SA is inherently sequential as each new solution depends on the previous one. The improvement of the parallel computational power has the ability to overcome this challenge.

In parallel computing, the problem under consideration is divided into several independent tasks that are executed simultaneously. Therefore, the execution time is expected to reduce. Obviously, more resources are needed as compared to the sequential environment. So, scalability and efficiency also increase. Three performance metrics are used to measure the effectiveness of a parallel algorithm; namely, the speedup, the efficiency, and the scalability. Speedup measures the extent of the execution time reduction of the parallel algorithm as compared to its sequential counterpart. The efficiency measures the resources utilization extent of the parallel program to the available hardware. Finally, the scalability measures the ability of the parallel program to handle larger problem sizes. So, the target of this research is to maximize the three previously mentioned metrics [7].

Since the main target is to generate a route plan for military drones emitted with the intention of acquiring images at multiple sites; time and energy are therefore critical factors. Thus, they are also taken into consideration in addition to finding a good solution.

This paper is structured as follows: Section 2 exposes the related work in the literature. Section 3 details the solution's design for both sequential and SMP algorithms. Section 4 introduces the results of both sequential and SMP versions. Finally, the conclusion and future work are summarized in Section 5.

2 Related Work

The drone route planning problem in this research is analogous to the TSP. There are several algorithms that provide a solution for the TSP, such as LS, BB, EAs, ACO, and hybrid algorithms [7]. This chapter exposes the sequential and parallel solutions to the TSP in general, and to the drone route planning in particular.

Many sequential algorithms are proposed to solve the TSP, some of which are based on the ACO algorithm. The proposed solution in [8] provides a modification of the traditional ACO method; this is known as the High Performance ACO. The traditional ACO algorithm involves a single ant randomly looking for the path; whereas the updated algorithm applies the TSP on a group of ants. The authors provide a comparison between their proposed algorithm and the ant colony system algorithm on various numbers of nodes. They found that the proposed algorithm completes the task in less time.

Also, Local search algorithms are widely used to solve the TSP. The research in [7] provides an experimental study to test the performance of the Lin-Kernighan and the Multi-Neighborhood Search. Results show that the Lin-Kernighan provides better results than the Multi-Neighborhood Search in terms of runtime.

On the other hand, several parallel solutions are proposed in the literature to solve the TSP using diverse parallel programming platforms. In [46], the experiment is performed on a standard multicore CPU. The reported results indicate that a speedup of 7.3 is gained on 8 cores. Thus, the usage of PSO algorithm is more suitable for real-time planning for the drone. Moreover, the experiments also proved that the performance of the GA is better than the PSO. The same authors improved their results in [9] by proposing a parallel hybrid algorithm that exploits the advantages of both the PSO and the GA to generate a suitable path plan for fixed-wing drones. It is found that the gained speedup is 10.7 on a 12-core SMP.

In [2], the authors planned the drone’s path using parallel ACO solution using a Graphical Processing Unit (GPU) platform. The generated path guides the drone in disseminating keys and collecting data from wireless sensors, which were previously deployed at minimum cost. The drone launches from a ground station, visits all sensors in a limited period of time, and then returns back to the ground station. In their experiment, the

researchers compared the sequential with the parallel performance. They showed that the speedup is higher when using GPU platform.

In [5], the authors generate multiple route paths for several drones simultaneously using synchronous parallel SA on the GPU. Experiments' results prove that the processing time is reduced and a better solution is acquired, as compared to the CPU implementation.

3 Design and Implementation

This section discusses the general design of SA. Then, the implementations in both sequential and SMP environments are discussed. In addition, the drone's energy manipulation is discussed.

3.1 Simulated Annealing Design

Two concepts are to be defined when it comes to designing an iterative metaheuristics algorithm; namely, the solution representation and the objective function. Since SA is classified as a single-solution based metaheuristics, it requires the definition of the neighborhood. These are detailed in the following subsections [10].

3.1.1 Solution Representation

The solution representation of the drone route planning is a permutation of size *n*, where *n* is the number of the waypoints to be visited exactly once. Each permutation represents one solution as shown in Figure 1 as a sequence of nodes; where each node represents a waypoint and its index represents the corresponding order. The number of all permutations that represent the solution space taking into consideration the fixed start point (ground station) is *(n-1)!*

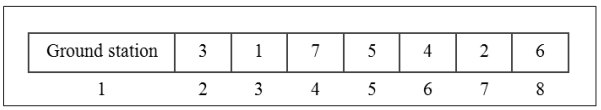


Figure 1. The permutation representation of the drone route plan problem

3.1.2 Neighborhood Solution

The neighborhood of a solution is found by performing a move operator which leads to a tiny perturbation to the solution *S* [10]. As the drone route plan is represented by a permutation, a neighborhood is generated by the swap operator between two elements in the solution as illustrated in Figure 2.

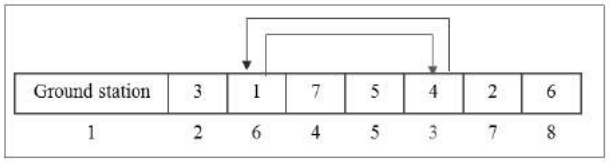


Figure 2. Neighbourhood solution generated by swapping the order of two waypoints

3.1.3 Objective function

The objective function is used to define the goal to be achieved by the SA. The goal of the problem in-hands is looking for the shortest route plan for a drone such that it visits each waypoint exactly once. As previously mentioned, this is similar to the TSP and has a similar objective function which is shown below:

$$f(\pi) = Min \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} (1)$$

where:
π is a permutation that represents a tour of the drone;
and *n* is the number of waypoints.

A. Sequential simulated annealing

In order to be able to measure the speedup of the parallel SA, the sequential implementation is to be developed. As mentioned previously, the SA, like other single-solution-based metaheuristics, includes two main steps. The first step is to generate the initial solution. The latter is constructed by either using a greedy heuristic, such as the nearest neighbor algorithm or randomly. In this research, the random method is used because the greedy heuristics produce a solution in the local optimum; this may not be able to provide an improved local optimum solution at the end [10].

The second step, which is the solution improvement, the design uses the swap operator between two points to generate a neighbor solution from the current solution.

The SA algorithm imitates the process of the solid hardening, which depends on the initial temperature value and the cooling rate. Therefore, the SA implementation consists of two main loops to provide a suitable solution. The outer loop, known as the cooling loop, is responsible for managing the temperature value. On the other hand, the inner loop, known as the equilibrium state loop, is responsible for constructing a

neighbor solution from the current one, evaluating it, and computing the probability of the acceptance using the following formula:

$$e^{(-\Delta/T)} \qquad (2)$$

where:
Δ is the difference between the cost of the old and the new solution;
T is the current temperature

Accordingly, the main parameters that are to be defined during SA implementation are the initial temperature, the cooling rate, and the stopping condition. The latter might be the minimum temperature or a specific number of iterations. In this research, the stopping condition is taken based on a minimum temperature. Usually, The other parameters are determined after several experiments [6]. The flowchart of the sequential algorithm is shown in Figure 3.

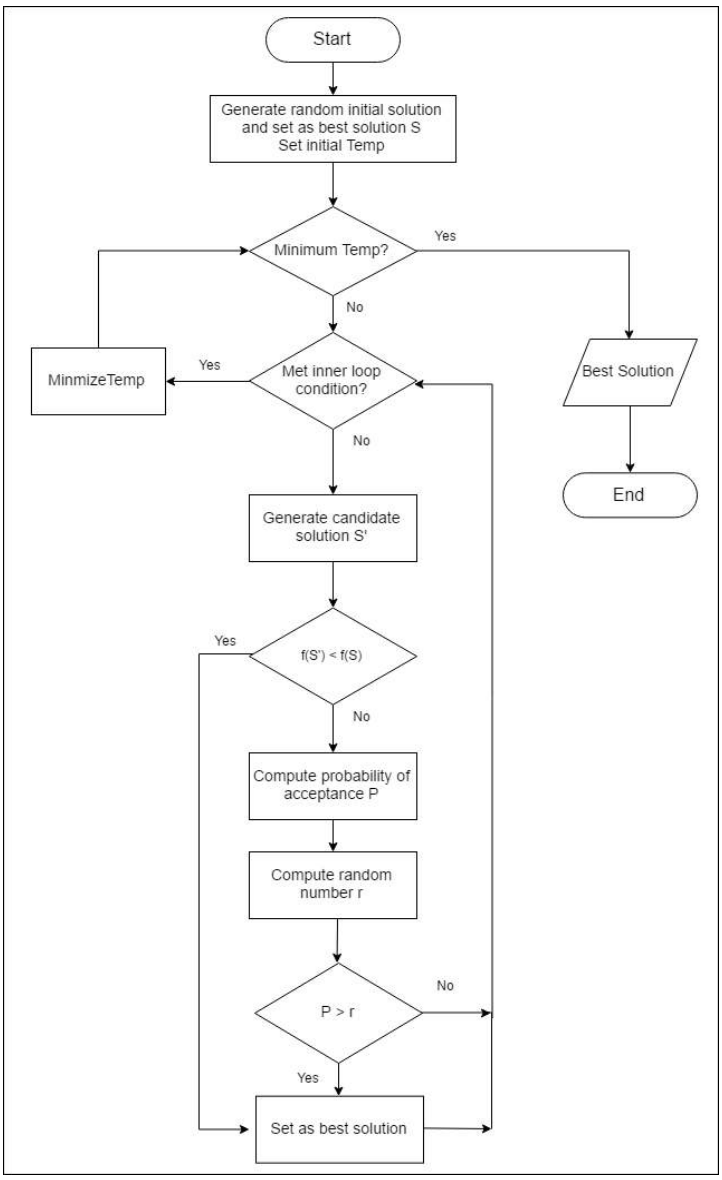


Figure 3. Flow chart of the SA algorithm

B. Parallel Simulated Annealing

Metaheuristic algorithms are sequential by nature; SA is no exception. Consequently, parallelizing the SA entails a challenging problem [11]. Many approaches are proposed to parallelize SA algorithm [12]. These are summarized below:

- Decompose the search space into smaller parts, and then assign each part to a processor to find the minimum cost. The results are then shared with other processors.

- Apply the asynchronous approach, where each processor executes SA independently. The initial solution may be either the same or different across the processors. Finally, compute a reduce operation to get the best solution among them. As illustrated in Figure 4.

- Apply the synchronous approach, where each processor uses the same initial solution and performs parallel improvements within the same temperature. Then at each temperature value, the best solution is shared between the processors to perform parallel improvement until the end. Figure 5 illustrates the synchronous approach.

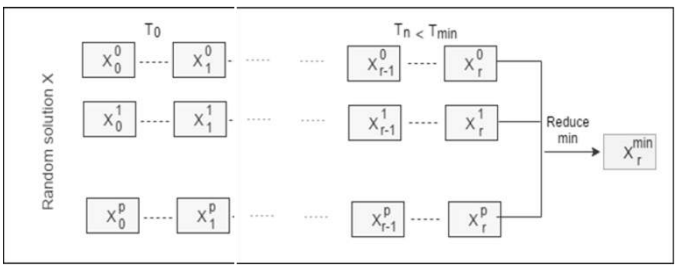


Figure 4. Asynchronous SA parallel approach

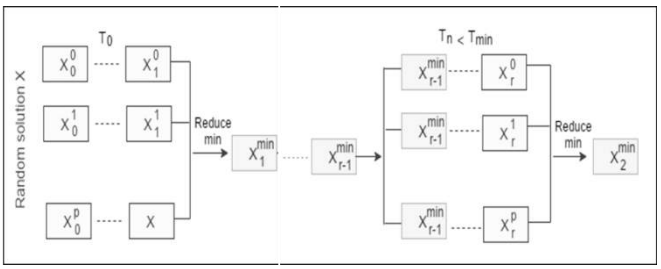


Figure 5. Synchronous SA parallel approach

In this study, synchronous SA parallel approach is used as it is able to parallelize the equilibrium state loop. The flowchart of the parallel SA algorithm is shown in Figure 6.

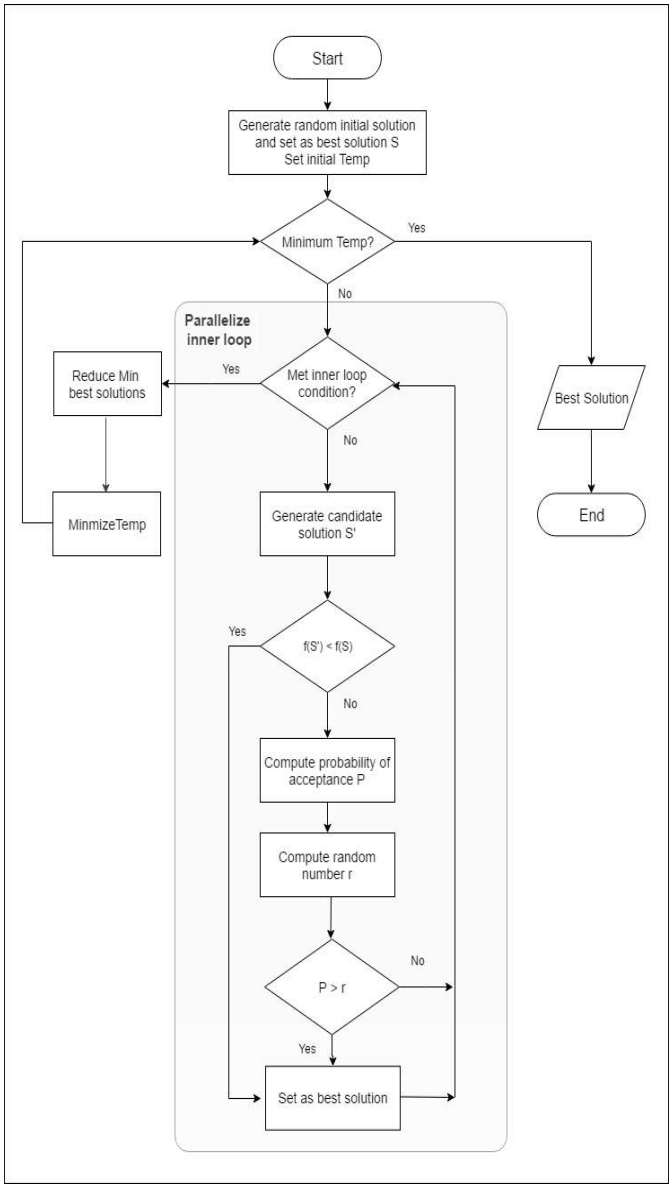


Figure 6. Flowchart of parallel SA approach

3.3 Handling the Drone Energy Constraints

After generating the route plan using SA algorithm at the ground station, the route is evaluated in terms of the energy required to complete the planned mission. If the energy level is above a predefined threshold, then the drone is emitted according to the planned route. Otherwise, the mission is divided into multiple journeys. The drone is re-charged after the end of each trip, before starting a new one.

In fact, the drone's energy is expressed in terms of its enduring lifetime L. Therefore, the time *T* needed to travel the final distance *D*, as planned by the SA, is calculated as follows:

$$T = D / S \qquad (3)$$

where:

- *T* is the time required to make the complete tour calculation, including the return trip to the ground station.
- *D* is the final distance as calculated by the SA algorithm
- *S* is the drone's speed as specified in its specifications

If the calculated time *T* is less than the drone's enduring lifetime *L*, then the drone is safely launched. Otherwise, the journey is broken into multiple trips. Figure 7 illustrates the previously mentioned steps.

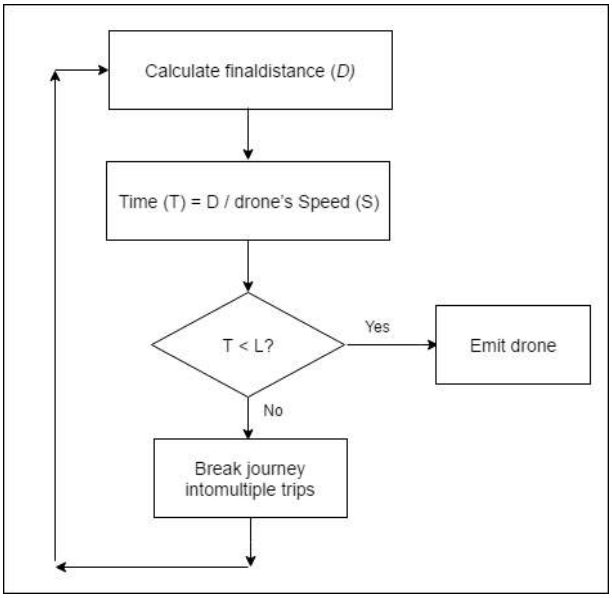


Figure 7. Checking drone's **energy**

Therefore, the applied predefined threshold is the enduring lifetime of the emitted drone.

4 Experiments and Result

This section details the methodology used in the conducted experiments. In addition, the obtained results are discussed, analyzed and represented graphically. The first subsection reveals the

deployed environment in terms of software and hardware specifications. Subsection 4.2 exposes the performance metrics used to measure the effectiveness of the developed algorithm. Finally, subsection 4.3 details the results of both the sequential and parallel algorithm. These are presented graphically, analyzed, and then discussed in detail.

4.1 Software and Hardware Specification

This research is implemented on an Intel quad-core i7, with a speed of 2.53 GHz, and supports hyperthreading. The program is coded in Java under Linux ubuntu 16. PJ2 is the used multithreading library. PJ2 manages the threads by taking care of synchronization and load balancing [11]. As a result, this minimizes the idle time of the processors, increases the utilization of the hardware resources. Obviously, this is one of the significant objectives in parallel computing [13].

4.2 Performance Measurements

The main objective of this research is to minimize the execution time, increase resources utilization, and increase scalability. These are measured by the speedup, the efficiency, and the sizeup respectively. The following subsections expose more details about performance measurements [11].

4.2.1 Speedup

Speed up is used to measure the extent of the time reduction gained from the parallel implementation as compared to its sequential counterpart. The gained speed up is calculated using the following formula [11, 14]:

$$\text{Speedup} = \frac{\text{Execution time of sequential version}}{\text{Execution time of parallel version}} \quad (4)$$

4.2.2 Efficiency

The efficiency (E) is used to measure how a program is close to the ideal status. In other words, it indicates the effectiveness of the resources utilization by the parallel program. The ideal program efficiency is equal to 1. However, the actual efficiency is between 0 and 1. Efficiency is directly proportional with the use of the available hardware resources, such as the number of cores/processors. Efficiency is computed using the following formula [11]:

$$\text{Efficiency} = \frac{\text{Actual Speed up}}{\text{Number of cores or processors}} (5)$$

4.2.3 Scalability

Scalability is the ability of a program to adapt to the increasing amount of problem size. Also, it is used to measure the ability of a program to making use of the increasing number of cores/processors. The scalability is calculated according to the following formula [11]:

$$\text{Scalability} = \frac{\text{Npar}}{\text{Nseq}} (6)$$

where:

- *Nseq* is the largest problem size that can be handled by the sequential program
- *Npar* is the largest problem size that can be handled by the parallel program for a specific number of threads.

It is expected that the problem size increases with the increase of the number of threads/processors.

4.3 Methodology

The execution time of a parallel program is hardly the same when run multiple times successively. This is because the operating system is conducting its own activities at the same time as the program runs. Since these activities differ from a run to another, the resulting execution time is directly affected. The interference of the operating system always increases the resulting execution time. Therefore, the following methodology is followed to measure the execution time of a program as accurate as possible: the program is run multiple times – from 7 to 10 times – and the execution time is recorded after each run. Then, the minimum of these recordings is taken since this represents the less interference from the part of the operating system. The speedup and the efficiency are then calculated according to formulae 4 and 5 respectively.

On the other hand, the following steps are performed to measure the scalability, or the sizeup. The sequential version of the program is run multiple times while increasing the problem size in each run. The largest problem size that can be handled by the sequential program, say *Nseq*, is then recorded. The same procedure is repeated for different number of threads to get *Ni,par* where *i* represents the used number of threads.

The sizeup efficiency is then calculated according to formula 6.

4.4 Experimental results

A set of experiments are conducted according to the methodology detailed in Section 4.3 with the aim of measuring the three main performance metrics: speedup, efficiency, and scalability in addition to the quality of the solution. The results of both versions of the SA algorithm are reported in the following subsections. The number of threads in the experiments range from 2 to 8.

4.4.1 Running Time Measurement

The first set of experiments aims to explore the impact of the number of waypoints on the execution time. Therefore, the program is run multiple times with various number of waypoints; namely, 50, 100, 250, and 500. For each problem size, the experiment is repeated as explained in Section 4.3. Table 1 in the Appendix shows the minimum execution time for each number of threads with the total route plan distance. Note that all the parameters of SA are fixed and chosen after several experiments to ensure the quality of the final solution. Figure 8 shows the impact of the number of waypoints on the execution time.

As seen in the graph depicted in Figure 8, the execution time of both sequential and SMP SA is proportional to the problem size. However, it is noticed that the increasing rate of the execution time of the sequential SA is higher as compared to that of the SMP SA for all threads. Table 2 in the Appendix details the increasing rates for each number of threads and for different problem sizes. Also, it is noticed from the graph that the execution time decreases with the increase of the number of threads. However, this rate decreases with the increasing number of threads.

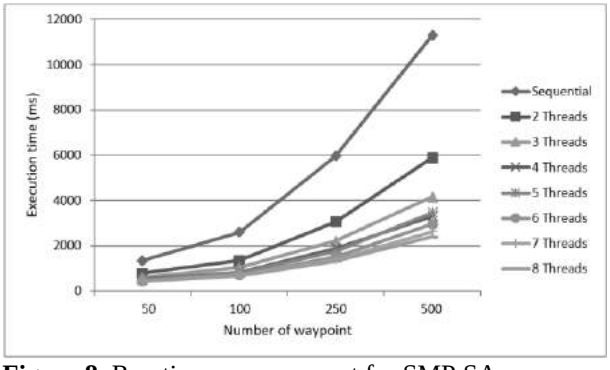


Figure 8. Run time measurement for SMP SA

4.4.2 Quality of the Route Plan

The final distance values are also depicted in Table 1 in the Appendix with various numbers of threads and waypoints. The corresponding graph is shown in Figure 9. It is noticed that the quality of the route plan is improved with the parallel execution for 81.25% of the reported results against degradation for 8.75% of the cases. Although the numbers of threads displayed on the x-axis are discrete in nature, however, the lines are represented continuous to illustrate the discrepancy between the recorded results. Also, it is noticed that the final distance increases with the increase of waypoints for all number of threads. This is expected since as, previously mentioned; the running time is directly proportional with the problem size.

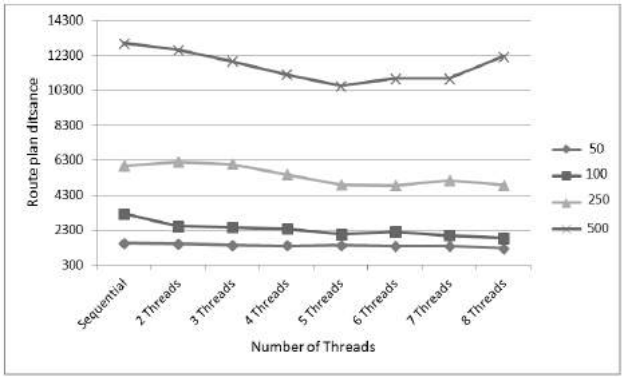


Figure 9. Quality of the final distance on SMP SA

4.4.3 Speedup Measurement

The gained speedup is calculated from the values recorded in Table 1 in the Appendix according to formula 4. Table 3 in the Appendix reports the calculations of the speedup for various numbers of cities and threads. Figure 10 depicts the same values graphically. It is noticed that the speedup is directly proportional to the number of deployed threads for all problem sizes. However, the increase rate of the speedup is less for higher values of thread numbers. These results are expected since the threads scheduling becomes more complicated for higher number of threads. Therefore, the imposed overhead is higher.

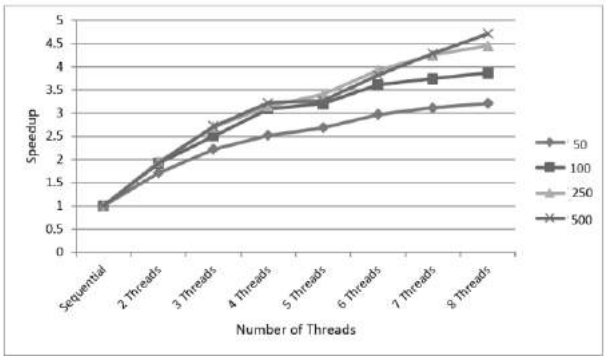


Figure 10. Speedup of SMP SA

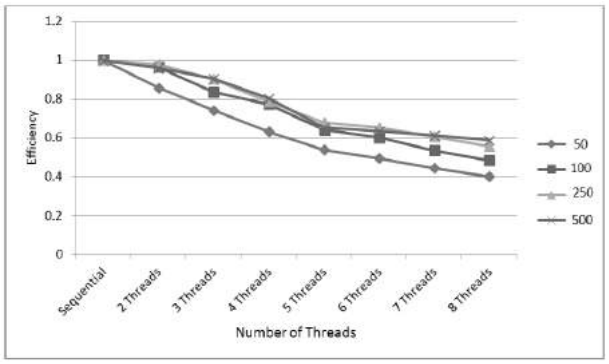


Figure 11. Efficiency of the SMP SA

4.4.4 Efficiency Measurement

As previously mentioned, the efficiency is measured according to formula 5. The results of the calculations for all numbers of threads and waypoints are depicted in Table 3 in the Appendix. The corresponding graph is depicted in Figure 11. It is noticed that the efficiency decreases with the increase of number of threads. This is expected since the actual speedup grows at a lower rate than that of the ideal speedup. This is due to the growing complexity of threads scheduling with higher number of threads. Therefore, the gap becomes larger.

Also, it is noticed that the efficiency is close to 1 for number of threads = 4. This is explained by the fact that the SMP on which the experiments are executed is a quad-core. Since the processor supports hyperthreading, it allows us to increase the number of threads to 8 on 4 cores. However, hyperthreading is expensive in terms of scheduling, and does not yield the same efficiency if compared with an 8-core processor. Also, it is noted from the graph that the efficiency tends to zero as the number of threads increases.

Worth to mention that although the efficiency is equal to 1 for the sequential program, but this does not imply that the program makes full use of the available hardware resources. However, the actual speedup of the sequential program and the number of cores are both equal to one.

4.4.5 Scalability Measurement

Since the hardware resources increase in an SMP, it is expected to handle larger problem sizes. The largest number of waypoints handled by each thread is divided by that handled by the sequential program. The results and calculations are reported in Table 4 in the Appendix. In addition, Figure 12 depicts the relationship between the maximum numbers of waypoints for each thread against the execution time. According to the results, the scalability is directly proportional to the number of threads. Also, it is noted that for the same number of waypoints, the execution time is reduced as the number of threads increases.

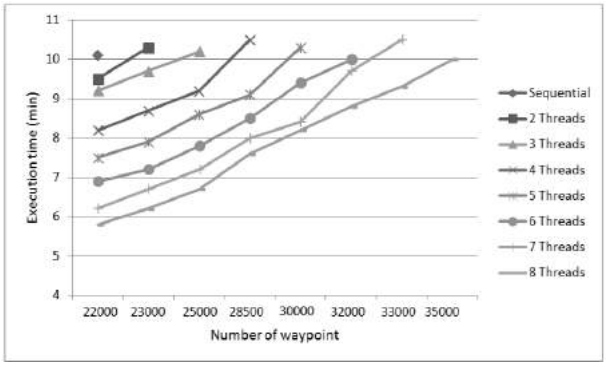


Figure 12. Scalability of the SMP SA

5 Conclusion and Future Work

This research targets for generating a minimum route plan distance for a single drone emitted by a military organism for image acquisition. The area in concern may be a sensitive site, a defense and/or attack war front, or an enemy's territory. Therefore, the route path should be completed in the least possible amount of time. The SA algorithm is implemented to solve this problem which is analogous to the TSP. Since metaheuristics require an extensively long time for execution, parallel computing is deployed to

accelerate the SA algorithm. However, one of the big challenges associated with parallel SA is the inherent sequential nature of the SA. This is each new solution depends on the previous one. This research implements parallel SA using parallel java library pj2 under Linux Ubuntu on an SMP. A quad-core laptop is used whose processor supports hyperthreading.

Three main performance metrics are used to measure the effectiveness of a parallel program; namely, the gained speedup, the efficiency, and the scalability. All metrics are to be maximized. The reported results prove that the gained speedup and the scalability increase with the increase of number of threads for all numbers of waypoints. On the other hand, the efficiency decreases with the increase of number of threads due to that the actual speedup does not grow at the same rate as the linear speedup. Furthermore, the quality of the SMP SA generated final solution is increased in 81.25% of the experiments.

In the future, the SA is to be implemented on KACST's SANAM supercomputer which embraces more than three-thousands processors. The gained speedup, the efficiency, the scalability and the quality of the route plan are investigated on the supercomputer. Results are to be compared with those of SMP and those reported in the literature.

REFERENCES

[1] N. Özalp and O. K. Sahingoz, "Optimal UAV path planning in a 3D threat environment by using parallel evolutionary algorithms," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference*, 2013, pp. 308-317.

[2] M. O. UgurCekmez, "A UAV Path planning with parallel ACO algorithm on CUDA platform," presented at the IEEE Unmanned Aircraft Systems (ICUAS), FL, USA, 2014.

[3] M. Coeckelbergh, "Drones, information technology, and distance: mapping the moral epistemology of remote fighting," *Ethics and information technology,* vol. 15, pp. 87-98, Jun 2013.

[4] X.-f. Liu, Z.-w. Guan, Y.-q. Song, and D.-s. Chen, "An optimization model of UAV route planning for road segment surveillance," *Journal of Central South University,* vol. 21, pp. 2501-2510, Jun 2014.

[5] T. Turker, G. Yilmaz, and O. K. Sahingoz, "GPU-Accelerated Flight Route Planning for Multi-UAV Systems Using Simulated Annealing," in *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, 2016, pp. 279-288.

[6] M. Sanjabi, A. Jahanian, S. Amanollahi, and N. Miralaei, "ParSA: parallel simulated annealing placement algorithm for multi-core systems," in *Computer Architecture and Digital Systems (CADS), 2012 16th CSI International Symposium*, 2012, pp. 19-24.

[7] Y. Wu, T. Weise, and R. Chiong, "Local search for the Traveling Salesman Problem: A comparative study," in *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, Beijing, China, 2015, pp. 213-220.

[8] S. K. Sahana and A. Jain, "High performance ant colony optimizer (HPACO) for travelling salesman problem (TSP)," in *International Conference in Swarm Intelligence*, 2014, pp. 165-172.

[9] V. Roberge, M. Tarbouchi, and F. ALLAIRE, "Parallel hybrid metaheuristic on shared memory system for real-time UAV path planning," *International Journal of Computational Intelligence and Applications,* vol. 13, p. 1450008, Jun. 2014.

[10] E.-G. Talbi, *Metaheuristics: from design to implementation* vol. 74: John Wiley & Sons, 2009.

[11] A. Kaminsky, "BIG CPU, BIG DATA: Solving the World's Toughest Computational Problems with Parallel Computing," 2016.

[12] A. Ferreiro, J. García, J. G. López-Salas, and C. Vázquez, "An efficient implementation of parallel simulated annealing algorithm in GPUs," *Journal of Global Optimization,* vol. 57, pp. 863-890, Nov. 2013.

[13] L. O. Mafteiu-Scai, "A parallel load balancing based on pseudo-cliques," *IJNCAA,* p. 94, 2015.

[14] S. Rastogi and H. Zaheer, "Significance of parallel computation over serial computation," in *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference*, 2016, pp. 2307-2310.

[15] E.-H. Guerrout, R. Mahiou, and S. Ait-Aoudia, "Medical image segmentation on a cluster of PCs using Markov random fields," *International Journal of New Computer Architectures and their Applications (IJNCAA),* vol. 3, pp. 35-44, 2013.

APPENDIX

Table 1 Minimum execution time of the serial and parallel SA with the route distance

#waypoints / #Threads	50		100		250		500	
	Time	**Distance**	**Time**	**Distance**	**Time**	**Distance**	**Time**	**Distance**
Sequential (1)	1351 ms	1565	2609 ms	3243	5976 ms	5997	11305 ms	13006
2	788 ms	1536	1355 ms	2531	3059 ms	6209	5900 ms	12634
3	607 ms	1442	1041 ms	2467	2210 ms	6072	4164 ms	11965
4	535 ms	1426	844 ms	2389	1896 ms	5478	3512 ms	11213
5	502 ms	1430	814 ms	2076	1758 ms	4913	3465 ms	10573
6	454 ms	1399	723 ms	2215	1523 ms	4867	2960 ms	10986
7	434 ms	1386	696 ms	2011	1404 ms	5148	2636 ms	10969
8	420 ms	1269	675 ms	1856	1342 ms	4875	2398 ms	12254

Table 2 Increase rate in the execution time for multiple threads and problem sizes

#waypoints / #Threads	50-100	100-250	250-500
Sequential (1)	93.1%	129 %	89.1 %
2	71.9 %	125.7 %	92.8 %
3	71.4 %	112.2 %	88.4 %
4	57.7 %	124.6 %	74.7 %
5	62.1%	115.9 %	85.2%
6	59.2 %	110.6 %	94.3%
7	60.3 %	101.7 %	87.7 %
8	60.7 %	98.8 %	78.6 %

Table 3 Measure performance of parallel SA on SMP (speedup-efficiency)

#waypoints	# Threads	Execution time (ms)	Speedup	Efficiency
50	Sequential (1)	1351 ms	1	1
	2	788 ms	1.714467	0.857234
	3	607 ms	2.2257	0.7419
	4	535 ms	2.525234	0.631308
	5	502 ms	2.691235	0.538247
	6	454 ms	2.975771	0.495962
	7	434 ms	3.112903	0.4447
	8	420 ms	3.216667	0.402083
100	Sequential (1)	2609 ms	1	1
	2	1355 ms	1.925461	0.962731
	3	1041 ms	2.506244	0.835415
	4	844 ms	3.091232	0.772808
	5	814 ms	3.20516	0.641032
	6	723 ms	3.608575	0.601429
	7	696 ms	3.748563	0.535509
	8	675 ms	3.865185	0.483148
250	Sequential (1)	5976 ms	1	1
	2	3059 ms	1.95358	0.97679
	3	2210 ms	2.704072	0.901357
	4	1896 ms	3.151899	0.787975
	5	1758 ms	3.399317	0.679863
	6	1523 ms	3.923835	0.653972
	7	1404 ms	4.25641	0.608059
	8	1342 ms	4.453055	0.556632
500	Sequential (1)	11305 ms	1	1
	2	5900 ms	1.916102	0.958051
	3	4164 ms	2.714938	0.904979
	4	3314 ms	3.218964	0.804741
	5	3465 ms	3.262626	0.652525
	6	2960 ms	3.819257	0.636543
	7	2636 ms	4.288695	0.612671
	8	2398 ms	4.714345	0.589293

Table 4 Measure scalability of parallel SA on SMP

# Threads	Size up	Scalability
Sequential (1)	22000	--
2	23000	1.05
3	25000	1.14
4	28000	1.28
5	30000	1.34
6	32000	1.46
7	33000	1.5
8	35000	1.6

International Journal of New Computer Architectures and their Applications (IJNCAA) 7(3): 89-94
The Society of Digital Information and Wireless Communications, 2017 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

A Bipolar Single Valued Neutrosophic Isolated Graphs: Revisited

Said Broumi
Laboratory of Information Processing, Faculty of Science Ben M'Sik, University Hassan II, B.P 7955,Sidi Othman, Casablanca, Morocco
broumisaid78@gmail.com

Assia Bakali
Ecole Royale Navale ,Boulevard Sour Jdid,B.P 16303 Casablanca, Morocco
assiabakali@yahoo.fr

Mohamed Talea
Laboratory of Information Processing, Faculty of Science Ben M'Sik, University Hassan II, B.P 7955,Sidi Othman, Casablanca, Morocco
taleamohamed@yahoo.fr

Florentin Smarandache
Department of Mathematics, University of New Mexico, 705 Gurley Avenue, Gallup, NM, 87301, USA
fsmarandache@gmail.com; smarand@unm.edu

Mohsin Khan
Department of Mathematics, Abdul Wali Khan University, Mardan 23200, Pakistan
mohsinkhan7284@gmail.com

ABSTRACT

In this research paper, the graph of the bipolar single-valued neutrosophic set model (BSVNS) is proposed. The graphs of single valued neutrosophic set models is generalized by this graph. For the BSVNS model, several results have been proved on complete and isolated graphs. Adding, an important and suitable condition for the graphs of the BSVNS model to become an isolated graph of the BSVNS model has been demonstrated.

KEYWORDS

Bipolar single valued neutrosophic graphs (BSVNG) , complete-BSVNG, isolated-BSVNGs.

1 Introduction

The concept of 'Neutrosophic logic' was developed by Prof. Dr. F. Smarandache in 1995 and get published in 1998. "It is a branch of philosophy which studies the origin, nature, and scope of neutralities, as well as their interactions with different ideational spectra"[4]. The concepts of fuzzy sets [8] and intuitionistic fuzzy set [6] were generalized by adding an independent indeterminacy-membership. Neutrosophic logic is a powerful tool to deal with incomplete, indeterminate, and inconsistent information, which is the main reason for widespread concerns of researchers. The concept of neutrosophic set(NS for short) is characterized by three independent degrees namely truth-membership degree (T), indeterminacy-membership degree (I), and falsity-membership degree (F).To practice NSs in real life situations efficiently,The subclass of the neutrosophic sets called single-valued neutrosophic set (in short SVNS) was defined by Smarandache in [4]. In another paper [5], Wang et al. defined the various operations and operators for the SVNS model. In [11] Deli et al. proposed a new concept called bipolar neutrosophic sets. This concept appear as a generalization of fuzzy sets, intuitionistic fuzzy sets, bipolar fuzzy sets, bipolar intuitionistic fuzzy sets and single valued neutrosophic set. The benefits of applying the NSs have been addressed in [18].The theory of graphs is the mostly used tool for resolving combinatorial problems in diverse disciplines like computer science, algebra and topology, etc. In [2, 4] Smarandache proposed two kinds of neutrosophic graphs to deal with situations in which there exist inconsistencies and indeterminacies among the vertices which cannot be dealt with by fuzzy graphs and different hybrid structures including bipolar fuzzy graphs, intuitionistic fuzzy graphs, bipolar intuitionsitc

fuzzy graphs [1,7,9, 10], The first kind is based on literal indeterminacy (I) component, the second kind of neutrosophic graphs is based on numerical truth-values (T, I, F), Recently, a hybrid study by combining SVNS and classical graph theory was carried out and that concept is called Single valued neutrosophic graph (SVNG) was presented by Broumi et al [12, 13, 14, 17, 20, 22].In addition, the concept of bipolar neutrosophic set was combined with graph theory and new graph model was presented. This concept is called bipolar single valued neutrosophic graph (BSVNGs). In [15,16] Broumi et al. proposed the concept of bipolar single valued neutrosophic graph as a generalized the concept of fuzzy graph, intuitionistic fuzzy graph, bipolar fuzzy graph and single valued neutrosophic graph.

The objective of this article is to demonstrate the essential and satisfactory condition of BSVNGs to be an isolated-BSVNG.

2.Background of research

Some of the important background knowledge in this paper is presented in this section. These results can be found in [4, 5, 12,13,15, 21].

Definition 2.1 [4] Le ζ be a universal set. The neutrosophic set A on the universal set ζ categorized into three membership functions called the true membership function $T_A(x)$, indeterminate membership function $I_A(x)$ and false membership function $F_A(x)$ contained in real standard or non-standard subset of $]^-0, 1^+[$ respectively and satisfy the following condition

$$^-0 \leq \sup T_A(x) + \sup I_A(x) + \sup F_A(x) \leq 3^+ \quad (1)$$

Definition 2.2 [5] Let ζ be a universal set. The single valued neutrosophic sets (SVNs) A on the universal ζ is denoted as following

$$A = \{ \langle x: T_A(x), I_A(x), F_A(x) \rangle x \in \zeta\} \quad (2)$$

The functions $T_A(x) \in [0.\ 1]$, $I_A(x) \in [0.\ 1]$ and $F_A(x) \in [0.\ 1]$ are called " degree of truth, indeterminacy and falsity membership of x in A", satisfy *the* following condition:

$$0 \leq T_A(x) + I_A(x) + F_A(x) \leq 3 \quad (3)$$

Definition 2.3 [12] A SVNG of $G^*= (V, E)$ is a graph G = (A, B) where

a. The following memberships: $T_A : V \rightarrow [0, 1]$, $I_A: V \rightarrow [0\ ,1]$ and $F_A: V \rightarrow [0, 1]$ represent the truth, indeterminate and false membership degrees of x ∈ V respectively and

$$0 \leq T_A(w) + I_A(w) + F_A(w) \leq 3 \quad (4)$$
$$\forall\ w \in V$$

b. The following memberships: $T_B: E \rightarrow [0\ ,1]$, $I_B: E \rightarrow [0, 1]$ and $F_B: E \rightarrow [0, 1]$are defined by

$$T_B(v, w) \leq \min [T_A(v), T_A(w)] \quad (5)$$

$$I_B(v, w) \geq \max [I_A(v), I_A(w)] \text{ and} \quad (6)$$

$$F_B(v, w) \geq \max [F_A(v), F_A(w)] \quad (7)$$

Represent the true, indeterminate and false membership degrees of the arc $(v, w) \in (V \times V)$, where

$$0 \leq T_B(v, w) + I_B(v, w) + F_B(v, w) \leq 3 \quad (8)$$
$$\forall\ (v, w) \in E$$

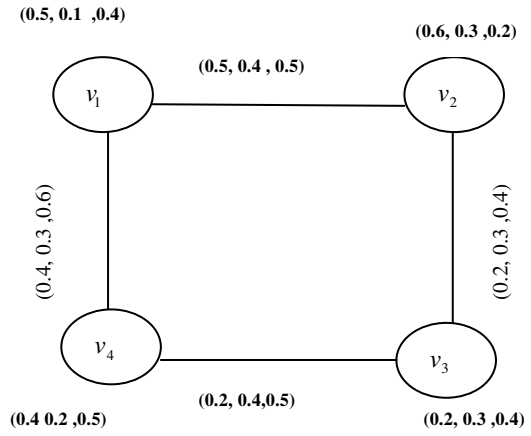


Fig.1.SVN-graph

Definition 2.4 [12]. A SVNG G= (A, B) is named a complete-SVNG if

$$T_B(v, w) = min [T_A(v) , T_A(w)] \quad (9)$$

$$I_B(v, w) = max[I_A(v) , I_A(w)] \quad (10)$$

$$F_B(v, w) = max [F_A(v) , F_A(w)] \quad (11)$$

$\forall\ v, w \in V$

***Definition* 2.5[12].** Let G = (A, B) be SVNG. Hence, the complement of SVNG G on G^* is a SVNG $\bar{G}$on G^* where

a. $\bar{A}$ = A (12)

b.$\overline{T_A}(w)=T_A(w)$, $\overline{I_A}(w) = I_A(w)$, $\overline{F_A}(w)=F_A(w)$ $\forall\ w \in V$ (13)

c.$\overline{T_B}\ (v, w) = \min [T_A(v),\ T_A\ (w)] - T_B\ (v, w)$ (14)

$\overline{I_B}\ (v, w) = \max [I_A(v),\ I_A\ (w)] - I_B\ (v, w)$ (15)

$\overline{F_B}(v, w)= \max [\ F_A(v), F_A(w)] - F_B\ (v, w)$, $\forall\ (v, w) \in E$.

Definition 2.6 [15].A BSVNG *G*= (A, B)of G^* = (V, E) is a partner such that A= (T_A^P, I_A^P, F_A^P,T_A^N ,I_A^N,F_A^N) is a BSVNS in V and B= (T_B^P,I_B^P, F_B^P,T_B^N ,I_B^N,F_B^N) is a BSVNS in E such that

(i) $T_B^P\ (v, w) \leq min\ (T_A^P(v),\ T_A^P(w))$ and $T_B^N(v, w) \geq max\ (T_A^N\ (v),T_A^N\ (w))$ (17)

(ii) $I_B^P(v, w) \geq max\ (I_A^P(v),I_A^P(w))$ and $I_B^N\ (v, w) \leq \min(I_A^N(v),I_A^N(w))$ (18)

(iii) $F_B^P\ (v, w) \geq max(F_A^P(v),F_A^P(w))$ and $F_B^N(v ,w) \leq \min(F_A^N\ (v),F_A^N\ (w))$, $\forall(v, w) \in E$ (19)

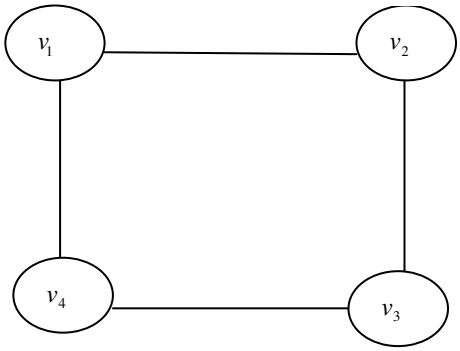


Fig.2 BSVNG

v_i	The values of vertex
v_1	(0.2, 0.2, 0.4 ,-0.4, -0.1,-0.4)
v_2	(0.1, 0.3, 0.5 ,-0.6, -0.2,-0.3)
v_3	(0.2, 0.3, 0.5 ,-0.3, -0.2,-0.1)
v_4	(0.3, 0.2, 0.4 ,-0.2, -0.3,-0.5)

Table1. The values of vertex of BSVNG

	The values of edge
v_{12}	(0.1, 0.3, 0.6 ,-0.2, -0.3,-0.1)
v_{23}	(0.1, 0.3, 0.6 ,-0.1, -0.6,-0.7)
v_{34}	(0.1, 0.5, 0.6 ,-0.1, -0.6,-0.5)
v_{14}	(0.2, 0.3, 0.5 ,-0.2, -0.3,-0.5)

Table2. The values of edge of BSVNG

Definition 2.7 [15]. The complement of BSVNG **G**= (*A*, *B*) of G^* =(V, E) is a BSVNG $\bar{G}$ = ($\bar{A}$, $\bar{B}$) of G^* =(V, E) such that

(i) $\bar{A}$ =A=(T_A^P,I_A^P, F_A^P,T_A^N ,I_A^N,F_A^N) and

(ii) $\bar{B}$ = ($T_{\bar{B}}^P$,$I_{\bar{B}}^P$,$F_{\bar{B}}^P$, $T_{\bar{B}}^N$,$I_{\bar{B}}^N$,$F_{\bar{B}}^N$) on E= V× V is defined as

$T_{\bar{B}}^P(v, w) = min\left(T_A^P(v),T_A^P(w)\right) - T_B^P(v, w)$

$T_{\bar{B}}^N\ (v\ , w)= max\left(T_A^N(v),T_A^N(w)\right) - T_B^N(v, w)$ (20)

$I_{\bar{B}}^P\ (v, w) = \max\left(I_A^P(v),I_A^P(w)\right) - I_B^P(v, w)$

$I_{\bar{B}}^N(v, w) = \min\left(I_A^N(v),I_A^N(w)\right) - I_B^N\ (v, w)$ (21)

$F_{\bar{B}}^P(v, w) = max\left(F_A^P(v),F_A^P(w)\right) - F_B^P(v, w)$

$F_{\bar{B}}^N\ (v, w)\ min\left(F_A^N(v),(w)\right) - F_B^N(v, w)$, (22)

$\forall\ \ (v,w)\in E$

Definition 2.8 [15]. A BSVNG *G*= (*A*, *B*) is called a complete-BSVNG if

$T_B^P\ (v, w) = \min\ (T_A^P\ (v),T_A^P\ (w))$, (23)

$T_B^N(v\ ,w) = max(T_A^N(v),T_A^N(w))$, (24)

$I_B^P(v, w)= max\ (I_A^P(v),I_A^P(w))$, (25)

$I_B^N(v, w) = min\ (I_A^N\ (v),I_A^N\ (w))$ (26)

$F_B^P(v, w) = max\ (F_A^P\ (v),F_A^P\ (w))$, (27)

$F_B^N(v, w) =min\ (F_A^N\ (v),F_A^N(w))$ (28) $\forall v, w\ \in V$

Definition 2.9[7]. The complement of BIFG **G**= (*A*, *B*) of G^* =(A, B) is a BIFG $\bar{G}$ = ($\bar{A}$, $\bar{B}$) of $\overline{G^*}$ =(V, V x V) where $\bar{A}$ =A=(T_A^P, F_A^P,T_A^N ,F_A^N) and $\bar{B}$ = ($\overline{T_B^P}$, $\overline{F_B^P}$,$\overline{T_B^N}$,, $\overline{F_B^N}$) are defined as

$\overline{T_B^P}(v, w) = min\left(T_A^P(v),T_A^P(w)\right) - T_B^P(v, w)$ (29)

$\overline{F_B^P}(v, w) = max\left(F_A^P(v),F_A^P(w)\right) - F_B^P(v, w)$ (30)

$\overline{T_B^N}\ (v\ , w)= max\left(T_A^N(v),T_A^N(w)\right) - T_B^N(v, w)$ (31)

$\overline{F_B^N}\ (v, w)\ min\left(F_A^N(v),(w)\right) - F_B^N(v, w)\ \ \forall v,$ w$\in$ V, $vw\in \tilde{V}^2$ (32)

Theorem 2.10[13]Let G =(A,B) be a SVNG, then the SVNG is called an isolated-SVNG if

and only if the complement of G is a complete-SVNG.

Theorem 2.11[21]Let G =(A,B) be a FG, then the FG is called an isolated-FG if and only if the complement of G is a complete- FG

3. MAIN RESULTS

Theorem 3.1: A BSVNG =(A,B) is an isolated-BSVNG iff the complement of BSVNG is a complete- BSVNG.

Proof: Let $G=(A, B)$ be a complete- BSVNG.

Therefore $T_B^p(v, w) = min\,(T_A^p(v), T_A^p(w))$,

$T_B^n(v, w) = max\,(T_A^n(v), T_A^n(w))$,

$I_B^p(v, w) = max\,(I_A^p(v), I_A^p(w))$,

$I_B^n(v, w) = min\,(I_A^n(v), I_A^n(w))$,

$F_B^p(v, w) = max\,(F_A^p(v), F_A^p(w))$,

$F_B^n(v, w) = min\,(F_A^n(v), F_A^n(w)), \forall v, w \in V$.

Hence in $\bar{G}$,

$\bar{T}_B^P(v, w) = \min(T_A^P(v), T_A^P(w)) - T_B^P(v, w)$

$= min(T_A^P(v), T_A^P(w)) - min(T_A^P(v), T_A^P(w))$

$=0$

and

$\bar{I}_B^P(v, w) = max\left(I_A^P(v), I_A^P(w)\right) - I_B^P(v, w)$

$= max(I_A^P(v), I_A^P(w)) - max(I_A^P(v), I_A^P(w))$

$= 0$

In addition

$\bar{F}_B^P(v, w) = max(F_A^P(v), F_A^P(w)) - F_B^P(v, w)$

$= max(F_A^P(v), F_A^P(w)) - max(F_A^P(v), F_A^P(w))$

$= 0$

We have for the negative membership edges

$\bar{T}_B^N(v, w) = max(T_A^N(v), T_A^N(w)) - T_B^N(v, w)$

$= max(T_A^N(v), T_A^N(w)) - max(T_A^N(v), T_A^N(w))$

$= 0$ and

$\bar{I}_B^N(v, w) = min(I_A^N(v), I_A^N(w)) - I_B^N(v, w)$

$= min(I_A^N(v), I_A^N(w)) - min(I_A^N(v), I_A^N(w))$

$= 0$

In addition

$\bar{F}_B^N(v, w) = min(F_A^N(v), F_A^N(w)) - F_B^N(v, w)$

$= min(F_A^N(v), F_A^N(w)) - min(F_A^N(v), F_A^N(w))$

$= 0$

So $(\bar{T}_B^P(v, w), \bar{I}_B^P(v, w), \bar{F}_B^P(v, w), \bar{T}_B^N(v, w), \bar{I}_B^N(v, w), \bar{F}_B^N(v, w)) = (0, 0, 0, 0, 0, 0)$

Hence G =(A, B) is an isolated-BSVNGs

Proposition 3.2: The notion of isolated-BSVNGs generalized the notion of isolated fuzzy graphs.

Proof: If the value of $I_A^P(w) = F_A^P(w) = T_A^n(w) = I_A^n(w) = F_A^n(w) = 0$, then the notion of isolated-BSVNGs is reduced to isolated fuzzy graphs.

Proposition 3.3: The notion of isolated-BSVNGs generalized the notion of isolated-SVNGs.

Proof: If the value of $T_A^n(w) = I_A^n(w) = F_A^n(w) = 0$, then the concept of isolated-BSVNGs is reduced to isolated-SVNGs.

Proposition 3.4: The notion of isolated-BSVNGs generalized the notion of isolated-bipolar intuitionistic fuzzy graph.

Proof: If the value of $I_A^p(w) = I_A^n(w)$, then the concept of isolated-BSVNGs is reduced to isolated-bipolar intutuitionistic fuzzy graphs

IV. COMPARTIVE STUDY

In this section, we present a table showing that the bipolar single valued neutrosophic graph generalized the concept of the crisp graph, fuzzy graph [9], intuitionistic fuzzy graph[1], bipolar fuzzy graph[10], bipolar intuitionistic fuzzy graph[7] and single valued neutrosophic graph[12].
For convenience we denote
F-graph : Fuzzy graphs

IF-graph: Intuitionistic fuzzy graph
BF-graph: Bipolar fuzzy graph
BIF-graph: Bipolar intutionistic fuzzy graph
SVN-graph: Single valued neutrosophic graph
BSVN-graph: Bipolar single valued neutrosophic graph

Type of graphs	The membership values of vertex/ edge					
	$T_A^p(w)$	$I_A^p(w)$	$F_A^p(w)$	$T_A^n(w)$	$I_A^n(w)$	$F_A^n(w)$
crisp graph	1 or 0	0	0	0	0	0
FG	$\in$ [0,1]	0	0	0	0	0
IFG	$\in$ [0,1]	0	$\in$ [0,1]	0	0	0
SVNG	$\in$ [0,1]	$\in$ [0,1]	$\in$ [0,1]	0	0	0
BFG	$\in$ [0,1]	0	0	$\in$ [-1,0]	0	0
BIFG	$\in$ [0,1]	0	$\in$ [0,1]	$\in$ [-1,0]	0	$\in$ [-1,0]
BSVNG	$\in$ [0,1]	$\in$ [0,1]	$\in$ [0,1]	$\in$ [-1,0]	$\in$ [-1,0]	$\in$ [-1,0]

Table3. Different types of graphs

Neutrosophic graph is the generalization of crisp graph, fuzzy graph, intuitionistic fuzzy graph, bi-polar fuzzy graph, bi-polar intuitionistic fuzzy graph and single-valued neutrosophic graph. In this table, we can see that by removing the indeterminacy and non-membership values from neutrosophic graph, the neutrosophic graph reduces to fuzzy graph. By removing the indeterminacy value from neutrosophic graph, the neutrosophic graph reduces to intuitionistic fuzzy graph. Similarly, by removing the positive and negative indeterminacy and non-membership values from bi-polar neutrosophic graph, the bi-polar neutrosophic graph reduces to bi-polar fuzzy graph. By removing the positive and negative indeterminacy values from bi-polar neutrosophic graph, the bi-polar neutrosophic graph reduces to bi-polar intuitionistic fuzzy graph. By the similar way, we can reduce a bi-polar single valued neutrosophic graph to a neutrosophic graph by removing the negative membership, indeterminacy and non-membership values.

5. CONCLUSION

In this article, we have proved necessary and sufficient condition under which BSVNGs is an isolated-BSVNGs. The notion of isolated-BSVNGs generalized the isolated-fuzzy graph and isolated- SVNGs. In addition, in future research, we shall concentrate on extending the idea of this paper by using the interval valued bipolar neutrosophic graph as a generalized form of bipolar neutrosophic graph.

REFERENCES

1. Gani, A. and Shajitha, B.S: Degree: Order and size in Intuitionistic Fuzzy Graphs. International Journal of Algorithms, Computing and Mathematics, 3(3) (2010).
2. Smarandache, F.: Refined Literal Indeterminacy and the Multiplication Law of Sub –Indeterminacies. Neutrosophic Sets and Systems, Vol.9, 58-63 (2015).
3. Smarandache, F.: Symbolic Neutrosophic Theory, Europanova asbl, Brussels, (2015)
4. Smarandache, F.: Neutrosophy, Neutrosophic Probability, Sets and Logic, Proquest Information & Learning, Ann Arbor, Michigan, USA, 105p,1998
5. Wang, H., Smarandache, F., Zhang, Y., and Sunderraman, R.: Single Valued Neutrosophic Sets. Multispace and Multistructure 4, 410-413 (2010).
6. Atanassov,K. : Intuitionistic Fuzzy Sets. Fuzzy Sets and Systems, Vol.20, 87-96 (1986).
7. Sankar, K., Ezhilmaran, D.: Balanced bipolar intuitionistic fuzzy graphs, International Research Journal of Engineering and Technology (IRJET),Volume: 03 Issue: 11, 806-812 (2016).
8. Zadeh, L.: Fuzzy sets. Information and Control, 8 338-35 (1965).
9. Bhattacharya, P.: Some Remarks on Fuzzy Graphs. Pattern Recognition Letters 6, 297-302 (1987).
10. Akram, M.: Bipolar Fuzzy Graphs. Information Science, doi:10.1016/j.ins.2011.07.037, (2011).
11. Deli, I., Ali, M., Smarandache, :Bipolar Neutrosophic Sets and Their Application Based on Multi-criteria Decision Making Problems, in: Advanced Mechatronic Systems (ICAMechS) 249- 254 (2015).
12. Broumi, S., Talea, M., Bakali, Smarandache, F.: Single Valued Neutrosophic Graphs. Journal of New Theory, N 10, 86-101(2016).
13. Broumi, S., Bakali, A., Talea, M., Smarandache, F.: Isolated Single Valued Neutrosophic Graphs. Neutrosophic Sets and Systems, Vol.11, 74-78 (2016).
14. Broumi, S., Talea, M., Smarandache ,F. and Bakali, A.: Single Valued Neutrosophic Graphs: Degree, Order and Size. IEEE International Conference on Fuzzy Systems, 2444-2451(2016).
15. Broumi, S., Smarandache, F., Talea, M. and Bakali, A.: An Introduction to Bipolar Single Valued Neutrosophic Graph Theory. Applied Mechanics and Materials , vol .841, 184- 191(2016).
16. Broumi, S., Talea, M., Bakali, A., Smarandache, F.: On Bipolar Single Valued Neutrosphic Graphs. Journal of New Theory, N11, 84-102 (2016).

17. Broumi, S., Bakali, A., Talea, M., Hassan, A., Smarandache, F.: Generalized Single Valued Neutrosophic Graphs of First Type,2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Gdynia Maritime University, Gdynia, Poland, 3-5 July ,413-420 (2017).
18. http://fs.gallup.unm.edu /NSS.
19. Hassan, A., Malik, M. A., Broumi, S., Bakali, A., Talea, M. and Smarandache, F.: Special Types of Bipolar Single Valued Neutrosophic Graphs, Annals of Fuzzy Mathematics and Informatics, Volume 14, No. 1, 55-73 (2017).
20. Broumi, S., Bakali, A., Talea, M., Smarandache, F.: Generalized Bipolar Neutrosophic Graphs of Type 1, 20th International Conference on Information Fusion Xi'an, China - July 10-13, 1714-1720 (2017).
21. Rahurikar, S. : On Isolated Fuzzy Graph, International Journal of Research in Engineering Technology and Management, Vol2,Issue 06, 1-3 (2014).
22. Broumi, S., Dey, A., Bakali, A., Talea, M., Smarandache, F., Son, L. H. and Koley D.: Uniform Single Valued Neutrosophic Graphs, Neutrosophic Sets and Systems, Vol. 17, 42-49 (2017)
23. Safar, M., Mahdi, F., and Mahdi, K., An Algorithm for Detecting Cycles in Undirected Graphs using CUDA Technology, International Journal on New Computer Architectures and Their Applications (IJNCAA) 2(1): 193-212 (2012)

International Journal of New Computer Architectures and their Applications (IJNCAA) 7(3): 95-101
The Society of Digital Information and Wireless Communications, 2017 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Development of an Algorithm based on Conservation of Energy and on a Hierarchical Routing Protocol

HAJAR LAGRAINI[1]
Radiations materials and instrumentations Laboratory University Hassan 1st, Faculty of Sciences and Technology FSTS, BP 577, 2600, SETTAT, MOROCCO
h.lagraini@uhp.ac.ma

MOSTAFA CHHIBA[1]
Radiations materials and instrumentations Laboratory University Hassan 1st, Faculty of Sciences and Technology FSTS, BP 577, 2600, SETTAT, MOROCCO
moschhiba@yahoo.com

ABDELMOUMEN TABYAOUI[1]
Radiations materials and instrumentations Laboratory University Hassan 1st, Faculty of Sciences and Technology FSTS, BP 577, 2600, SETTAT, MOROCCO
tabyaoui@uhp.ac.ma

AHMED MOUHSEN[2]
Management, industrial and innovation Laboratory
University Hassan 1st, Faculty of Sciences and Technology FSTS, BP 577, 2600, SETTAT, MOROCCO
mouhsen.ahmed@gmail.com

Abstract

Wireless Sensor Networks are a rising innovation for attraction of scientist researchers with its examination challenges and different application areas. It consists of smart sensor nodes with detecting, calculation, and wireless exchanges capacities. The restricted vitality asset is one of the fundamental worries confronting the development of such systems. The purpose of this paper is to develop an algorithm for energy optimization and lifetime enduring based on Low Energy Adaptive Clustering Hierarchy (LEACH) protocol. The new protocol selects a cluster head (CH) node by bearing in mind geometric space between the nominated nodes and the sink, additionally by observing remaining energy of the node better than the remaining energy level of nodes in the WSN. Simulation results show that our proposed algorithm outperformed the basic LEACH in terms of preserving energy such that stretching network lifetime.

Keywords—IoT; WSN; LEACH; Clustering; smart sensors; Data transmission.

Introduction

With the current advances in electromechanical appliances and wireless communication technology a large variety of ease and low power utilization smart sensors are accessible. Sensors gather the information from its encompassing territory, complete straightforward calculations, and communicate with rest of sensors or with the base station (BS) [1].

Primary difficulties of wireless sensors system are energy efficiency, and the system lifetime because as we all know the battery capacity of the smart sensor is restricted and due to tough deployment of the sensor nodes, it is unreasonable to charge or recharge the battery.

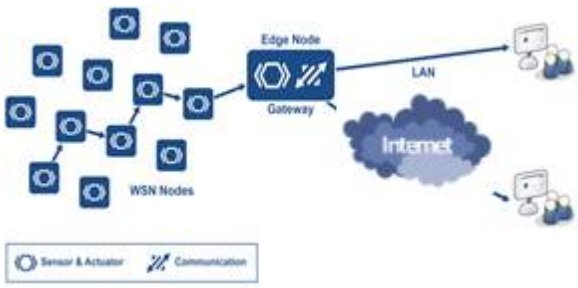


Figure 1 : Architecture of a wireless sensor network

Therefore clustering of the network is an unconventional technique to elevate the energy efficiency and performance of the network. In hierarchical architecture sensing area is separated in to a number of clusters and a set of nodes are intermittently chosen as (CH).CH assembles the information from non-CH members, combines it and then forwards it to the base station for further treating [2]. Clustering consequently dispense the energy load, decrease the energy depletion and surge the network existence [3], [4], [5], [6]. In this work a novel energy efficient cluster based system is proposed for WSNs. The approach proposes a new cluster head (CH) selection technique based on two factors: energy and distance. This protocol is a value-added of pure LEACH protocol presented in [7] and simulation result shows that the proposed algorithm is improved than LEACH. This paper is organized as follows. Section II briefly revisits the background of the basic LEACH algorithm as well as its derivatives. In section III, we discuss LEACH enhancements based on the distance and energy. And we evaluated the proposal comparatively in section IV. Finally, the conclusions and perspectives are discussed in section V.

RELATED WORKS

A. LEACH OVERVIEW (Low-Energy Adaptive Clustering Hierarchy)

LEACH (Low Energy Adaptive Clustering Hierarchy) is the most popular hierarchal routing protocol in WSN. It is a hierarchal routing protocol that uses the mechanism clustering. LEACH [7] was first introduced by Heinzelman in 2000. The defined operation of LEACH can be alienated into two parts: Set up phase, in which CHs are organized, and steady state phase, in which data are diffused to the sink.

The Threshold is set as follow:

$$T(n) = \begin{cases} \frac{p}{1-p\times(r \bmod \frac{1}{p})} & ;if\ n \in G \\ 0 & ;otherwise \end{cases} \quad (1)$$

However in this equation (1), p is the probability of selecting nodes as a CH node, r is the number of the current round, $(rmod\frac{1}{p})$ is on behalf of the number of SNs which was designated as CH in the round r, G is defined as the set of SNs that have not been CHs in the last ($\frac{1}{p}$) r.

As it is mentioned above the process of LEACH protocol is divided into rounds and each round is subdivided into two segments namely as:

- Set up phase:

An arbitrary number is created between ‘zero’ and ‘one’ by each node. This number is then compared with the Threshold value T (n). If this number is less than T (n), node is chosen as the CH. The elected node proclaims itself as a CH by sending a broadcast message to the whole network. Each node resolves which cluster to connect according to received signal strength and transmits a request message to the contiguous CH. CH announces them as the members of clusters on receiving all the messages by nodes.

- Steady state phase:

All the members of the clusters transfer information to the CH. Then CH collects the collected data and forwards this bonded data to the BS. The conventional LEACH has lot of witnesses. LEACH algorithm picks the CHs arbitrarily. While electing the CH it does not take in account the remaining of energy level of the SN. Thus if a node having the lower energy is designated as the CH, it may fall short of energy of the node and henceforth demise. Thus this node will become invalid. As the number of dead nodes growth there will be a damaging impact on network performance which will definitely reduce the network battery life.

LEACH protocol has the following advantages:

- The hierarchy, routing data and path selection are rather simple, and the SNs do not need to store large amounts of routing information, and do not need complex functions. The CH node is haphazardly designated, the chance of each node is the same, and the load of the WSN is balance [8].

However, there are a number of weaknesses in LEACH protocol, such as:

- For the reason that the CHs in LEACH protocol are arbitrarily generated, energy Consumption can be consistently distributed

in the network. However, the CH alternations do not take into account the residual energy of SNs. Therefore, it is possible for a sensor node that has low left over energy to be designated as a CH. This can provide the cluster useless due to the rapidly shattered battery power of the CH.

- In addition, LEACH does not analyze the distance between SNs and the BS. Consequently, if the geographic position of the CH is far from the sink and the geographic position of the cluster members is far from their CH, it will devour a lot of energy transmitting and receiving data. Subsequently it leads to exhaust the energy speedily in SNs [8].

B. DERIVATIONS OPTIMIZATION OF LEACH

Most of the proposed derivations enhanced LEACH by adding energy factor to the threshold criteria equation T (n), for example, M-LEACH [9], A-LEACH [10], LEACH-HEM [11], LEACH-C [12]

M-LEACH [9] solves the problem for the nodes having the lesser energy of becoming the CH. This growths the probability of nodes having the more left over energy to become the CH and therefore rises the survival time successfully. Threshold is defined as:

$$T(n) = \begin{cases} \frac{p}{1-p\times(r\ mod\frac{1}{p})} \times \frac{Ecurr}{Einit} & ;if\ n\ \in G \\ 0 & ;otherwise \end{cases} \quad (2)$$

Where, E_{curr} is the current energy of the node and E_{init} is defined as node's initial energy. Depending upon the energy consumption node is nominated as CH. Node which consumes less energy will be selected as cluster head.

A-LEACH [10] proposes a new CH selection algorithm that allows choosing the greatest node for CH. A-LEACH forms clusters by using a distributed Algorithm, where nodes make autonomous decisions without any centralized control. The node with more residual energy has a bigger probability to be a CH as follows:

$$T(n) = \begin{cases} \frac{K}{N-K\times(r\ mod\frac{N}{K})} \times \frac{Ecurr}{Emax} \times \frac{K}{N} & ;if\ n\ \in G \\ 0 & ;otherwise \end{cases} \quad (4)$$

HEM-LEACH [11] controls the probability of CH based on remaining energy. Greater the value of residual energy, upper is the probability of becoming the CH. If (r) is the reference energy compared with residual energy of node then,

$$Pi = p_{opt} \times \left[\frac{\bar{E}(r)-Ei(r)}{\bar{E}(r)}\right] = p_{opt} \times \frac{Ei(r)}{\bar{E}(r)} \quad (5)$$

$P_{opt}$is defined as the optimal percentage of CHs (P_{opt} is replaced by Pi), $\bar{E}$ (r) is the average energy of the network in round 'r', andthreshold for each node in each round is given as follow:

$$T(Si) = \begin{cases} \frac{p_i}{1-p_i\times(r\ mod\frac{1}{p_i})} & ;if\ Si\ \in G \\ 0 & ;otherwise \end{cases} \quad (6)$$

LEACH-C [12] LEACH-centralized is an improvement over LEACH algorithm. All the nodes transmit their energy details to the BS and then the BS transfer its result of which nodes are chosen as a CH. The sink select CH based on the residual energy of the SNs. Leach –C performance degrades when the energy charge for communicating with the base station is more than the energy cost for CH formation.

PROPOSED ALGORITHM

On analyzing the properties of LEACH, the nodes elected as cluster heads are not an ideal choice for transmitting the data to sink. In order to solve this problem, this section presents an efficient algorithm to compromise a significant saving in energy consumption. The proposed routing algorithm is appropriate for a wireless sensors network which has ensuing specifications: sensors are fixed, i.e., when they are first haphazardly distributed, they stay in the same place. Each node has an exclusive identifier and knows its current position and its remaining energy.

Furthermore, each node has sufficient power to communicate directly with the BS. All nodes have the same initial energy (E_{init}) and their batteries are not rechargeable or replaceable. SNs die with the end of their battery.

In this work, the probability for a node to become a CH depends on its distance to the sink. However attenuation of sending power decreases exponentially with the increasing diffusion distance in wireless communication. The value of residual energy of nodes and the distance of CHs to the BS are imperative metrics which have an impact on entire energy consumption in each round. Hence, we need to ponder the choice of the CH so that the range of CHs to the sink and distance of cluster members to the cluster head are lowest, which will significantly optimize energy consumption for WSN battery life improvement.

C. OPTIMIZATION MODEL

In our algorithm a sensor node is more expected to be chosen as a CH based on its distance from the BS and the residual of energy, the threshold value is calculated as follow:

$$T(n) = \begin{cases} C \times (1-b) \times \frac{E_{curr}}{E_{init}} + b * p * \left[\frac{D_{max}-D}{D_{max}+D}\right] & ; if\ n \in G \\ 0 & ; otherwise \end{cases} \quad (7)$$

While; C= $\frac{p}{1-P\times(r\ mod\frac{1}{p})}$

Where, E_{curr} is the residual energy of the sensor node for the current round and E_{init} is the initial energy, According to this algorithm, CH is selected based on distance of SN and BS. The node which is nearer to the base station will be elected as the cluster head. This enhancement takes remaining energy and distance into account, further it considers the distance from node to cluster head BS and compares the distance from node CH and BS. D_{max}: Represents the maximum distance from SN to BS. D represents distance from sensor node to the BS. Based on left over of energy and distance from BS, threshold criteria T(n)is calculated.

RESULTS AND SIMULATION

In order to evaluate performance and lifetime of the proposed algorithm compared to the basic LEACH, we used MATLAB. We set up a simulation scenario with the parameters presented in the following table. For simulation, 100 nodes are placed arbitrarily in a bounding environment. The initial energyE_{init} is similar for all the SNs.

Table 1 : Simulation Environment Specifications

Parameter	*Value*
Simulator	MATLAB
WSN area / m^2	100x100
Number of nodes	100
Number of rounds	3000
Initial energy of node : Ei	0.5 Joules
Desired percentage of CHs : P	0.05
Transmitter Amplifier Energy { Emp / Efs	0.0013 pJ/bit/m4 / 10 pJ/bit/m2
Data Aggregation Energy E_{agg}	5 nJ/bit
Electric energy	50 nJ/bit

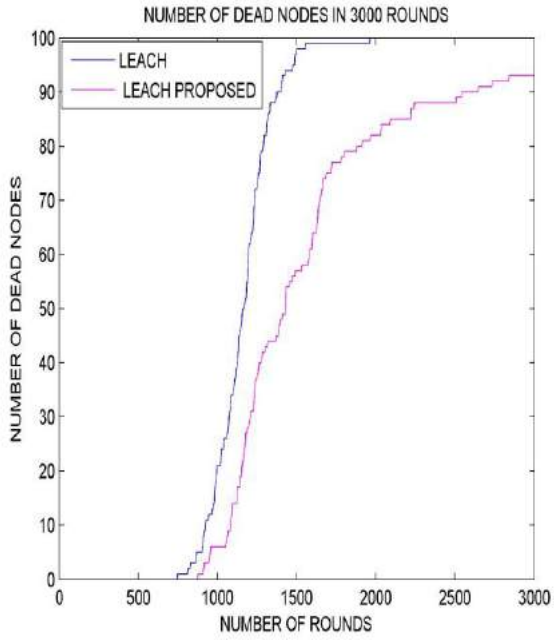


Figure 2 : Number of nodes alive in 3000 rounds

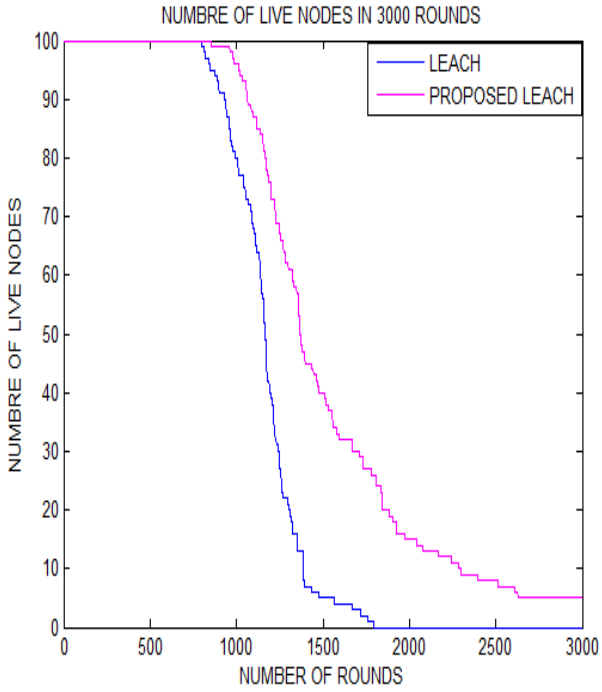


Figure 3 : Number of dead nodes in 3000 rounds

The results are shown in Figure 2; nodes in the proposed LEACH have a longer network battery life than the basic LEACH. Here, the Y axis designates the number of SNs and the X axis indicates the lifetime of the SNs in the wireless sensors network according to the simulation time. According to Figure 3, proposed LEACH is better than basic LEACH in terms of network lifetime. The reduction in power ingesting of proposed LEACH is measured by comparing it with that of the basic LEACH algorithm. We observe from the figure that the nodes of basic algorithm start to die approximately at 720 rounds. However, the proposed LEACH nodes start to die at 960rounds. We can also clearly notice that the network lifetime of proposed algorithm prolonged until 3000 rounds, whereas that of basic LEACH ended at 1800 rounds.

This result shows the capability of proposed LEACH to improve power consumption.

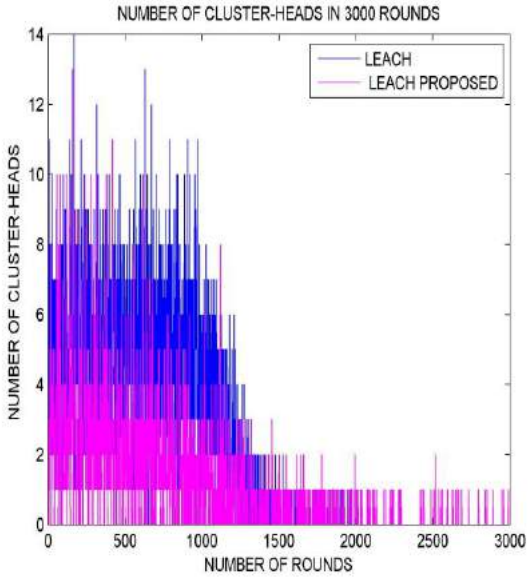


Figure 4: Number of CH created in 3000 rounds

In Fig. 4 the number of created CHs in 3000 rounds of the enhanced protocol is so considerably greater than the number of generated CHs in 3000 rounds using the pure formula of CHs election in LEACH; also in Fig. 5 data packet transportation is higher for our suggested formula .

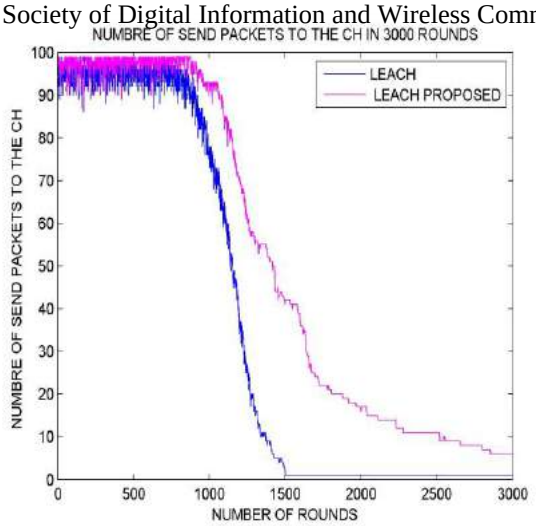


Figure 5: Number of CH created in 3000 rounds

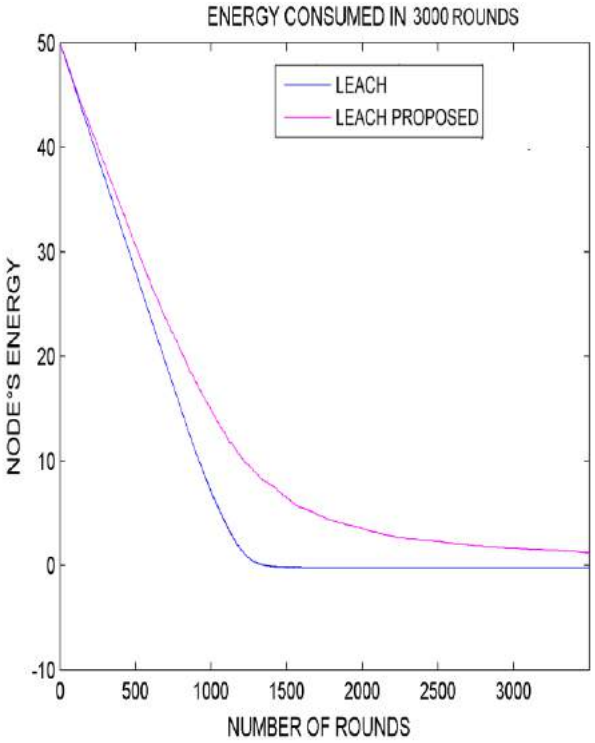


Figure 6: Energy ingesting by pure and new LEACH algorithm

Figure 6 compares the energy ingesting between the pure and modified algorithm in 3000 rounds. The energy consumed based on the operations performed, such as sensing operations, data treating and communication. It is crystal clear that new LEACH wastes less energy than the pure LEACH.

CONCLUSIONS AND FUTUR WORK

In this work, we propose an enhancement to LEACH algorithm for better energy efficiency in WSN, we devise a new technique to choose the cluster heads in every round which depends both on distance and residual of energy. In our modified algorithm, we consider obviously that the CH nearby the BS will dissipates less energy than other SNs because communication of data consumes the most energy in WSNs.

Simulation results confirmed that the modified LEACH outperforms pure LEACH in extending WSN battery life. However this work can achieve performance improvements, still many aspects of its design need to be explored since there are other metrics that may influence the energy consumption, comprehensive simulation and analysis could be additionally examined. Moreover there are other areas for improvement to make our developed LEACH best suited everywhere. As the smart sensor nodes have limited power energy, therefore the nodes die after a certain period limit.

Future work directions are to take the nodes themselves as solar aware nodes which recapture energy it selves, so that our protocol will be more energy sufficient. And another provision is to make our algorithm in to hierarchal protocol by forming ”TOP clusters” out of the cluster head nodes and TOP clusters will process all the information from the CHs. Therefore the energy dissipation of every CH will be condensed to transfer data to the sink. This enhancement will make the modified LEACH effective for an extensive range of wireless smart sensor networks.

REFERENCES

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. Computer networks, 38(4), 393-422.

[2] Akkaya, K., & Younis, M. (2005). A survey on routing protocols for wireless sensor networks. Ad hoc networks, 3(3), 325-349.

[3] Al-Karaki, J. N., & Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. IEEE wireless communications, 11(6), 6-28.

[4] Liang, C. K., Huang, Y. J., & Lin, J. D. (2008, March). An energy efficient routing scheme in wireless sensor networks. In Advanced

Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on (pp. 916-921). IEEE.

[5] Zheng, J., & Jamalipour, A. (2009). Wireless sensor networks: a networking perspective. John Wiley & Sons.

[6] Lindsey, S., & Raghavendra, C. S. (2002). PEGASIS: Power-efficient gathering in sensor information systems. In Aerospace conference proceedings, 2002. IEEE (Vol. 3, pp. 3-3). IEEE.

[7] Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. IEEE Transactions on wireless communications, 1(4), 660-670.

[8] Bian, Q., Zhang, Y., & Zhao, Y. (2010, May). Research on clustering routing algorithms in wireless sensor networks. In Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on (Vol. 2, pp. 1110-1113). IEEE.

[9] Long-long, X., & Jian-jun, Z. (2010, August). Improved LEACH cluster head multi-hops algorithm in wireless sensor networks. In Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on(pp. 263-267). IEEE.

[10] Ali, M. S., Dey, T., & Biswas, R. (2008, December). ALEACH: Advanced LEACH routing protocol for wireless microsensor networks. In Electrical and Computer Engineering, 2008. ICECE 2008. International Conference on (pp. 909-914). IEEE.

[11] Chen, G., Zhang, X., Yu, J., & Wang, M. (2012, July). An improved LEACH algorithm based on heterogeneous energy of nodes in wireless sensor networks. In Computing, Measurement, Control and Sensor Network (CMCSN), 2012 International Conference on (pp. 101-104). IEEE.

[12] Ahmad, A., Latif, K., Javaidl, N., Khan, Z. A., & Qasim, U. (2013, May). Density controlled divide-and-rule scheme for energy efficient routing in Wireless Sensor Networks. In Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on (pp. 1-4). IEEE.

International Journal of New Computer Architectures and their Applications (IJNCAA) 7(3): 102-111
The Society of Digital Information and Wireless Communications, 2017 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Hybrid Design Space Exploration Methodology for Application Specific System Design

K.Balasubadra[$], A.P.Shanthi[#], and V. Prasanna Srinivasan[*]

[$]Professor, Dept of Information Technology, R.M.D Engineering College, E-Mail: kbalasubadra@gmail.com
Professor, Dept of Computer Science & Engg, College of Engineering, Anna University, E-Mail: apshanthi@annauniv.edu
*Associate Professor, Dept of Information Technology, R.M.D Engineering College, E-Mail: vas.sri81@gmail.com (Corresponding Author)

ABSTRACT

During the application specific system design process, the designer has to take early decisions for selecting the optimal system components from the available huge design alternatives. To obtain the optimal design configuration from the available design alternatives, an efficient Design Space Exploration (DSE) process is required. This paper extends our previous work by integrating Bayesian Belief Network (BBN) based design space pruning methodology with the proposed heuristic algorithm for appropriate selection of memory configuration during the system design process. A complete and well structured DSE strategy has been formulated through the combination of BBN and heuristic approaches. The BBN performs design space pruning from the available huge design alternatives, resulting in near Pareto-optimal solution. The proposed heuristic algorithm performs the selection of the most optimal cache options. This paper mainly focuses on integrating the BBN with the proposed heuristic algorithm for providing efficient DSE strategy that aids the system designers during the application specific system design process. The experimental results in support of the proposed heuristic show a considerable reduction in the number of simulations required for covering the design space and also the algorithm finds the most optimal cache configurations for the given application with less number of iterations.

Keywords:

APPLICATION SPECIFIC SYSTEM DESIGN, DESIGN SPACE PRUNING, DESIGN SPACE EXPLORATION, HEURISTIC ALGORITHM

1. INTRODUCTION

The embedded computing system has transformed itself into a ubiquitous system because of its usage, ranging from simple applications to high end high performance systems. This transformation has been possible because of the advancements in the Electronic System Level (ESL) design. The rapid growth in the development of electronic devices forces the designer to offer not only multipurpose devices with high performance but also with longer battery life, low cost and availability of the product for usage in shorter time period. Therefore, the designer has to select the most optimal configuration from the available huge design options, along with conflicting criteria in order to meet the design goals. In order to obtain the optimal design configuration from the available huge design alternatives, an efficient Design Space Exploration (DSE) process is required. The architectural parameters affecting the overall objectives of the system should be considered during the design, so that the search process for finding the optimal system configurations will be rapid and more efficient.

The stringent application requirements such as low power design, low energy and more functionality with lesser cost have forced the

designers of such systems to optimize the design towards application specific design. During the design of such application specific systems, one of the primary choices that have to be made is the memory hierarchy design. The designer has to choose a better cache configuration option that suits the application. Since the number of memory accesses depends on the application, each application requires its own cache configuration option to attain the best performance. Every memory access will be accounted for in the overall system power consumption. Therefore, choosing the optimal cache configuration for a particular application is very important.

This paper describes a heuristic algorithm to support the designers during cache configuration exploration and integrates it with the BBN based design space pruning approach, to choose the most optimal cache option and to further reduce the simulation time.

2. RELATED WORK

Random sampling is employed usually for exploring the design space. Simulated annealing, Genetic Algorithms and Tabu search are the common techniques that are used in the literature for providing exploration heuristics.

A multi – objective DSE using genetic algorithm has been proposed by Palesi & Givargis [1] for efficiently exploring a parameterised System on Chip (SoC) architecture to find all Pareto optimal configurations in a multi-objective design space. The methodology uses a parameter dependency model of the target architecture to extensively prune non-optimal design spaces. Locally, the approach applies genetic algorithm to discover Pareto optimal configurations within the remaining design points. Heuristic based cache configuration exploration strategy for prototyping platform and its hardware implementation has been described by Zhang and Vahid [2].

Srinivasan et al [3] compare explorations driven by simulated annealing with results using an evolutionary approach. The work described by Shee et al [4] consider streaming application which is manually partitioned into a series of algorithmic stages and formulates a heuristic to efficiently search the design space for a pipeline based multi-Application Specific Instruction Processor (ASIP) system. Anderson & Khalid [5] have proposed a design space exploration framework, which performs initial configuration sweep for collecting architectural information, and using genetic algorithm for the design space exploration.

A Generalized Linear Model (GLM) and genetic algorithm based approach for efficiently exploring bus-based communication architectures of SoCs has been proposed by Esmeraldo & Barros [6]. The GLM is a statistical method based on the technique of Maximum Likelihood, in which parameters of a generalized linear model can be estimated when the errors follow an exponential family distribution. In order to train the model, sufficient number of simulation test results is needed. To reduce the number of training sets needed, genetic programming has been employed.

Javaid et al [7] have proposed a novel methodology of exploring the Application Specific Instruction Processors (ASIP) design space. The methodology first builds a pipeline of processors targeted for executing streaming applications. Initially, a heuristic has been used to rapidly explore a large number of processor configurations to find near Pareto Front of the design space, and then an Exact Integer Linear Programming Formulation (EIF) has been used to find an optimal solution. A Reduced ILP Formulation (RIF) has been used if the EIF does not find an optimal solution in the given time window.

An efficient two step DSE framework has been proposed for high performance SoC based on synchronous data flow specification by Lee et al [8]. In the first step, the mapping avoidance and pinning-first co-synthesis algorithms allow the execution of multiple tasks. In the next step, the Pareto-optimal set of on chip bus architectures has been obtained by using static performance analysis and trace-driven simulation. This is a recent work that considers the exploration of both the architecture and communication design spaces. Three different approaches such as multi-objective genetic algorithm, multi-objective Simulated Annealing, and multi-objective Tabu search have been used by Ceriani et al [9] for simultaneously exploring the architecture, mapping and scheduling of the system considering multi-rate real-time applications with certain objectives to be optimised such as area, hard and soft deadlines, and dimensions of memory buffers.

A new online design space exploration algorithm - CAPS has been proposed by Liu et al [10] which iteratively invokes a commercial tool (the oracle) to synthesise various instances of the components and implicitly builds approximations for their Pareto sets. Ascia et al [11] compare the performance of various Multi Objective Evolutionary Algorithms (MOEA) considering real-time applications for demonstrating the scalability and accuracy of the approaches. Pomata et al [12] have introduced a methodology enabling the use of FPGA based prototyping for micro-architectural DSE of ASIPs. To increase the emulation speed-up, the proposed technique exploits the translation of application binary code compiled for a custom ASIP architecture, into code executable on a different configuration.

Reliability aware DSE has been proposed by Huang et al [13] considering both the temporal and spatial redundancy. The work also considers both the transient and intermittent faults with imperfect fault detectors during the reliability analysis. Multi-objective evolutionary algorithm has been implemented for incorporating the results of the reliability analysis for performing automated DSE. A novel design space exploration strategy for autonomous High Level Synthesis (HLS) under resource constraints has been proposed by Prost-Boucle [14]. This methodology focuses on the rapidity of the HLS flow by using both the iterative and greedy strategy for the exploration process. The experimental results show the potential capability of the approach in speeding up of the generation of the hardware accelerators for Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuit (ASIC).

A Design space exploration strategy based on meta-heuristic has been proposed by Nikitin et al [15] for Chip Multiprocessor (CMP). Analytical model has been implemented that incorporates the contention of hierarchical interconnects, by resolving cyclic dependencies between memory latency and traffic. Finally, the design configurations are reduced by using the meta-heuristic exploration, during the exploration process.

The motivation to carry out this work is that the system designer has the responsibility of selecting the most optimal configuration for an application along with conflicting criteria, which poses a harder challenge during the system design process. The proposed technique aims to reduce the burden of the system designer during the selection of the most optimal cache configuration for a given application. The problem definition and description of the proposed technique is presented in the next section.

3. PROBLEM OVERVIEW

Given a specification of the application and system requirements, the designer needs to shrink the range

of feasible designs to a smaller number of possible designs that are best suited for the given application. The design space can increase exponentially with respect to the number of processing elements and other micro-architectural parameters considered, and thus increase the complexity of the search process. Consequently, a disciplined approach to the DSE is needed in order to evaluate large design spaces with high number of potential designs. This includes algorithms to prune and cover the design space in a systematic way at different levels of abstraction and refinement.

In order to obtain the optimal design configuration from the available huge design alternatives, an efficient design space pruning technique that will ease the DSE process is required. The knowledge about the architectural parameters affecting the overall objectives of the system should be considered during the design, so that the search process for finding optimal system configurations will be rapid and more efficient.

Since considerable amount of time is needed for the selection of optimal design, the most promising effort considered is to provide a mechanism to aid the designer for selecting the optimal design configuration from the available alternatives, considering multiple objectives and to take early design decisions. The challenge for the designer is how to incorporate domain knowledge of the architectural parameters during the search of optimal design options from the available huge design space.

Both fine grained and coarse grained exploration methodology are highly essential during the complex system design process. The fine grained exploration strategy concentrates on the internals of the individual components. For example, customizing the instruction set of processor architecture to meet the application requirement. The coarse grained exploration strategy instead concentrates on the system level, considering different possible processing elements, communication architectures and memory options.

The following aspects should be considered during the development of an efficient DSE strategy:

- Any DSE strategy should guide the system designer in taking early design decision.
- The implemented DSE strategy should provide an unbiased selection of design points.
- The DSE strategy should ensure that all the recommended design points have been evaluated for their correctness.
- Validation of the final design, which ensures that all the required objectives have been satisfied.

Thus, it is evident that the development of an efficient DSE technique is highly essential during the design of any complex application specific systems.

A hybrid DSE strategy is proposed based on Bayesian Belief Network (BBN) modeling framework [19] and heuristic search methodology. The proposed strategy attempts to resolve the existing limitation in imparting domain knowledge and provides a pioneering effort to support the designers during the process of application specific system design. Figure 1 shows the high level design space exploration flow of the proposed methodology. The application type, the objectives to be satisfied and the architectural parameters are the input to the proposed BBN model. The construction, validation and the utilization of the proposed BBN based DSS framework for design space pruning forms the key aspect during the DSE flow.

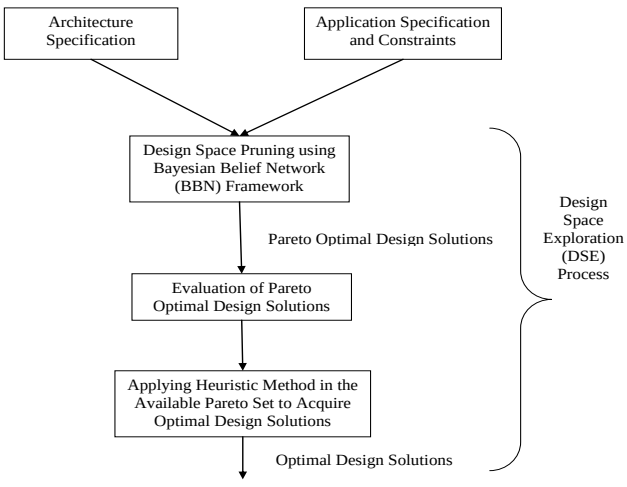


Figure 1: Proposed DSE Flow

The architectural specification, application specification and constraints forms as inputs to the DSE flow. The design space pruning is performed through the BBN based modeling framework. The introduction to BBN, the obtained results of the pruning process and the quantitative comparison of those results with the state-of-the-art techniques has been described in our previous work [19]. The design space pruning using the BBN results in providing near Pareto-optimal solution for the given application.

Further, to explore the available cache configurations within those Pareto-optimal solutions, several numbers of simulations is still required to identify the most optimal cache configuration for the given application. The goal is to search for the most optimal cache configuration options that are suitable for the application within the near Pareto-optimal solutions provided by the BBN, along with the constraints that should be satisfied. The constraints that should be satisfied during the search process require that the execution time, energy and power should be minimal for the corresponding design options.

The three important cache parameters that should be considered during cache exploration are cache size, associativity and line size. Cache misses will be lesser if the cache size is high, but at the same time cost increases for the larger cache size because of larger silicon and it will also consume more power due to the higher static and dynamic power dissipation.

Cache associativity is the other important parameter that determines how many locations in a cache are simultaneously searched for a given address. In a direct mapped or one way set associative cache, only one tag and data location is searched. In a two way set associative cache, two tags and two data locations are simultaneously searched. One way, two way, four way and eight way set associative caches are commonly used. Direct-mapped caches are fast and have low power per access, but have very low hit rates and hence very poor performance. Adding associativity to four ways results in good performance, but at the cost of slower time per access and much more power per access.

Another important cache parameter is the line size, which is the number of bytes brought into the cache, whenever there is a cache miss. The cache parameter options will not be the same for all the applications. Some applications require larger number of memory accesses and some other applications may not require that many. Therefore, choosing the most optimal cache options is important during the system design process, also considering the conflicting criteria.

Generally, during exploration, a set of optimal configurations will be obtained for a particular application, which is known as the Pareto-optimal set. A configuration is part of the Pareto-optimal set if no other configuration which is not part of the Pareto set has better solution. The proposed heuristic algorithm will search through various cache configurations, for finding the most optimal Pareto set which has minimal execution time, energy and power.

3.1 Proposed Heuristic Algorithm

Input: IC size, DC size, cache Associativity, and cache line size.

Output: The best set of configurations which has minimum values for Execution Time (ET), Energy (E) and Power (P).

1. Initialize the set min-values{min_ET ← 0, min_E ← 0, min_P ← 0}
2. Initialize the set max-values{max_ET ← 0, max _E ← 0, max _P ← 0}
3. Initialize i ← 1, j ← 1 and k ← 1
4. for i = 1 to m do
5. for j = 1 to n do
6. for b = 1 to r do
7. Simulate_Config(IC[i], DC[j], A[k], l[b])
8. update the values of the sets min-values and max-values

9. pareto set (Yes) ← configurations which has any one minimum value
10.obtain min_config[] ← configurations which has at least one minimum value
11.end for
12. k = k+1
13.flag = true
14. b=1
15.for each min_config[] do
16. while(flag=true)
17. Simulate_Config(IC[i], DC[j], A[k], l[b])
18.if obtained any one minimum value in the set min-values then
19. update the values of the sets min-values and max-values
20.flag=true
21.b=b+1
22. pareto set(Yes) ← configuration which has the minimum value
23. else
24. obtained any one maximum value in the set max-values
25.flag=false
26. pareto set (No) ← configuration which has the maximum value
27. end if
28.end while
29.end for
30.i=i+1, j=j+1
31. end for
32. end for

4. EXPERIMENTAL RESULTS

The Xtensa customizable processor architecture [16] from Tensilica is considered as an example target platform for performing the experimental simulations. The Xtensa platform offers various sizes of Instruction Cache (IC) and Data Cache (DC), cache associativity and line size to be chosen by the designer for a particular application. Table 1 describes the total available options of the cache configurations alone from the Xtensa architecture.

Table 1: Total Cache Configuration design space options available in the target architecture

Memory Options	Available Configuration	Example	Available Design Space
Instruction Cache (IC)	1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K	Need to choose 'IC' and 'DC' from totally 8 options, so 8^2 = 64 options. Need to choose 'AS' and 'L' from totally 3 options, so 3^2 = 9 options.	$8^2 \times 3^2 = 576$
Data Cache (DC)	1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K		
Associativity (AS)	Direct Mapped, 2-Way, 4-Way		
Line Size (L)	16-Byte, 32-Byte, 64-Byte		

It is difficult and time consuming to explore all the 576 combinations of the configuration for choosing the best option for a given application. The experimental study is performed considering computationally intensive, memory intensive and communication intensive application workloads. The ALPbench [17] and Mediabench [18] benchmark suite provides various application workloads for analyzing the performance of the target architecture. Three different applications such as Joint Photographic Experts Group (JPEG) encoder, Motion JPEG encoder, and Advanced Encryption Standard (AES) have been considered for the performance analysis. This experimental validation helps to justify how the proposed heuristic algorithm is utilized for selecting the most optimal cache configuration option. In order to explain the algorithm clearly, this experiment considers that the Instruction Cache (IC) and Data Cache (DC) sizes vary from 32K to 128K, cache associativity as direct mapped, two way and four way and line size as 16, 32 and 64 bytes. The total available option now becomes $3^2 \times 3^2 = 81$.

The AES application is simulated using the Xtensa architecture simulator for the available 81 different cache configurations for the purpose of validating the proposed heuristic algorithm. The execution time, energy and power values for each of

the configurations are recorded for the purpose of comparison. Table 2 describes each step of the proposed algorithm for the AES application benchmark. Figure 2 show the values of the execution time, energy and power obtained in each of the simulated configurations for the AES application benchmark.

The algorithm searches for the optimal configurations during each iteration and reduces the number of simulations considerably during the search process. For the considered AES benchmark only forty seven simulations are necessary and thirty eight of them form the optimal Pareto set. Figure 3 show the comparison of results for various benchmarks. The result shows that the proposed heuristic algorithm finds the optimal configurations with lesser number of simulations. The percentage reduction in the number of simulations is 42%, 43% and 47% respectively, for the corresponding AES, MPEG and JPEG benchmark applications.

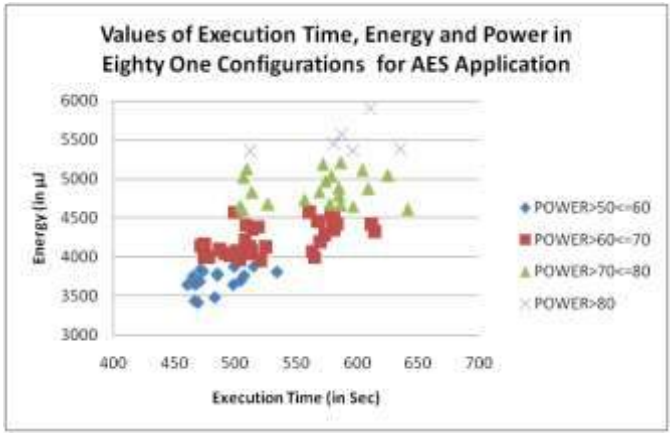


Figure 2: Values of Execution time, energy and power in Eighty One Configurations for AES application.

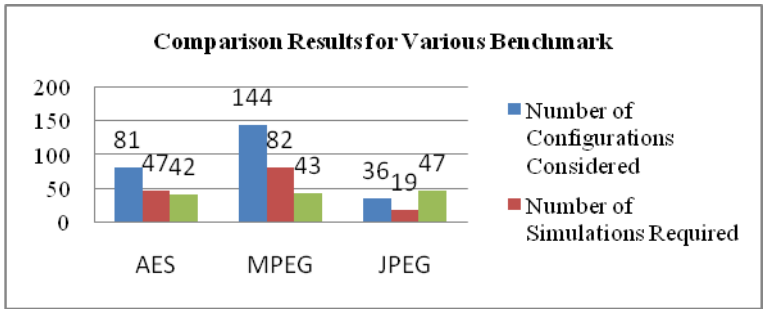


Figure 3: Comparison Results for Various Benchmarks.

5. CONCLUSION

The hybrid DSE methodology utilizing BBN and a heuristic algorithm is proposed in this paper, which searches for the most optimal design configuration that is suitable for a given application. The pruned design space with respect to a case study using the proposed method has shown a significant reduction in the number of design options and therefore number of simulations. The percentage reduction in the required number of simulations for the corresponding AES, MPEG and JPEG benchmark applications is 42%, 43% and 47% respectively. The observation that can be made from the obtained result is that the proposed methodology has the ability to guide the search process towards optimal design points, thereby reducing the number of simulations drastically for evaluating each design point.

Table 2: Step by step implementation of the proposed algorithm.

Number of Iterations	Number of Simulations	Configuration (IC,DC,Asso,LS)	Execution Time (Seconds)	Energy (Joules)	Power (Milli Watt)	Obtained Any Minimum or Maximum Value	Included within Pareto Set (Yes/No)
1	1	128,128,DM,64	**474.9**	**3998.9**	**60.9**	Min-(ET,E,P) Max-(ET,E,P)	Yes
	2	128,128,DM,32	***513.3***	4051	61.6		Yes
	3	128,128,DM,16	475.2	***4169.3***	***63.4***		Yes
	4	128,128,TWO,64	506.5	5019.8	76.4		No
2	5	128,64,DM,64	***564.8***	**3995.4**	**60.8**	Min-(E,P) Max-(ET)	Yes
	6	128,64,DM,32	510.6	4034.2	61.4		Yes
	7	128,64,DM,16	525.1	4125.5	62.8		Yes
	8	128,64,TWO,64	561.1	4568.4	69.5		No
3	9	128,32,DM,64	485	**3763**	**57.2**	Min-(ET,E,P)	Yes
	10	128,32,DM,32	**473.3**	3820.5	57.8		Yes
	11	128,32,DM,16	498.9	3877.6	59		Yes
	12	128,32,TWO,64	614.6	4321.5	65.8		No
4	13	64,128,DM,64	476.8	3993.1	60.8	NIL	Yes
	14	64,128,DM,32	491.1	4036.6	61.47		Yes
	15	64,128,DM,16	472.2	4131.8	62.9		Yes
	16	64,128,TWO,64	499.9	4575.9	69.6		No
5	17	64,64,DM,64	478.4	3989.7	60.7	NIL	Yes
	18	64,64,DM,32	496.5	4019.8	61.2		Yes
	19	64,64,DM,16	487.6	4088	62.2		Yes
	20	64,64,TWO,64	511.8	4124.5	62.8		Yes
	21	64,64,TWO,32	507.6	4208.9	64.1		No
6	22	64,32,DM,64	**465.3**	3757.2	**57.1**	Min-(ET,P)	Yes
	23	64,32,DM,32	485.4	3786.1	57.6		Yes
	24	64,32,DM,16	471.6	3840.1	58.4		Yes
	25	64,32,TWO,64	514.7	3877.6	59		Yes
	26	64,32,TWO,32	520.8	3947.2	60.1		Yes
	27	64,32,TWO,16	513.6	4083.4	62.1		Yes
	28	64,32,FOUR,64	573.2	4273.9	65		No
7	29	32,128,DM,64	467.6	**3648.5**	**55.5**	Min-(E,P)	Yes
	30	32,128,DM,32	471	3687.8	56.1		Yes
	31	32,128,DM,16	507.3	3770.5	57.4		Yes
	32	32,128,TWO,64	556.8	4733.1	72		No
8	33	32,64,DM,64	**460.2**	**3644.9**	**55.4**	Min-(ET,E,P)	Yes
	34	32,64,DM,32	463.6	3670.9	55.8		Yes
	35	32,64,DM,16	467.4	3726.6	56.7		Yes
	36	32,64,TWO,64	499.5	3891.5	59.7		Yes
	37	32,64,TWO,32	503.6	3958.4	60.2		Yes
	38	32,64,TWO,16	501	4069.7	61.9		Yes
	39	32,64,FOUR,64	580.7	4340.9	66.1		No
9	40	32,32,DM,64	469.3	**3412.5**	**51.9**	Min-(E,P)	Yes
	41	32,32,DM,32	466.6	3437.2	52.3		Yes
	42	32,32,DM,16	483.2	3478.8	52.9		Yes
	43	32,32,TWO,64	498.4	3644.6	55.5		Yes
	44	32,32,TWO,32	504.5	3696.7	56.3		Yes
	45	32,32,TWO,16	534	3805.1	57.9		Yes
	46	32.32,FOUR,64	562.6	4073.4	62		Yes
	47	32,32,FOUR,32	570	4186.7	63.7		No

Minimum values in the table during each iteration are represented as bold and maximum values in each iteration are represented in italics.

REFERENCES

1. Palesi M., Givargis T.: Multi – Objective Design Space Exploration Using Genetic Algorithms. In: *Proc. IEEE. Symposium on Hardware/Software Co-Design*, pp. 67-72. (2002).
2. Zhang C., Vahid F.: Cache Configuration Exploration on Prototyping Platforms. In: Proc of International Conference on Rapid System Prototyping, pp. 164-171, (2003).
3. Srinivasan V., Radhakrishnan S., Yemuri R.: Hardware Software Partitioning with Integrated Hardware Design Space Exploration. In: *Proc. Int. Conf. Design Automation, Test in Europe, pp.* 28-35, (1998).
4. Shee S.L., Erdos A., Sri Parameswaran.: Architectural Exploration of Heterogeneous Multiprocessor Systems for JPEG. *Int J Parallel Process*. 36, 140-162, (2008).
5. Anderson I.D.L., Khalid M.A.S.: SC build: a Computer – Aided Design Tool for Design Space Exploration of Embedded Central Processing Unit Cores for Field Programmable Gate Arrays. *IET Comput Digit Tech*. 3, 24-32, (2008).
6. Esmeraldo G., Barros E.: A Genetic Programming Based Approach for Efficiently Exploring Architectural Communication Design Space of MPSoC's. In: *Proc. IEEE. Int. Conf. Programmable Logic Conference*. pp. 29-34, (2010).
7. Javaid H., Ignjatovic A., Sri Parameswaran.: Rapid Design Space Exploration of Application Specific Heterogeneous Pipelined Multiprocessor Systems. *IEEE T Comput Aid D*. 29, 1777-1789, (2009).
8. Lee C., Kim S., Ha S.: A Systematic Design Space Exploration of MPSoC Based on Synchronous Data Flow Specification. *J Signal Process Syst*. 58, 93-113, (2010).
9. Ceriani M., Ferrandi F., Lanzi P.L., Sciuto D., Tumeo A.: Multiprocessor System on Chip Synthesis Using Multi – Objective Evolutionary Computation. In: *Proc. Int. Conf. Genetic and Evolutionary Computation Conference*. pp. 1267-1274, (2010).
10. Liu H.Y., Diakonikolas I., Petracca M., Carloni L.: Supervised Design Space Exploration by Compositional Approximation of Pareto Sets. In: *Proc. Int. Conf. Design Automation*. Pp. 399-404, (2011).
11. Ascia G., Catania V., Nuovo A.G.D., Palesi M., Patti D.: Performance Evaluation of Efficient Multi – Objective Evolutionary Algorithms for Design Space Exploration of Embedded Computer Systems. *Appl Soft Comput*. 11, 382-398, (2011).
12. Pomata S., Meloni P., Tuveri G., Raffo L., Lindwer M.: Exploiting Binary Translation for Fast ASIP Design Space Exploration on FPGA's. In: *Proc. Int. Conf. Design Automation, Test in Europe*. pp. 566-569, (2012).
13. Huang J., Raabe A., Huang K., Buckl C., Knoll A.: A Framework for Reliability-Aware Design Exploration on MPSoC Based Systems. *Des Autom Embed Syst*. (2013).
14. Prost-Boucle., Muller O., Rousseau F.: Fast and standalone Design Space Exploration for High-Level Synthesis under resource constraints. Journal of System Architecture. 60, 79-93, (2014).
15. Nikitin N., de San Pedro J., Cortadella J.: Architectural Exploration of Large-Scale Hierarchical Chip Multiprocessors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 32 (10), 1569-1582, (2013).
16. Xtensa Processor. Tensilica Inc (2011) Available from. http://www.tensilica.com/
17. Li M., Sasanka R., Adve S., Chen Y.K., Debes E.: The ALPBench Benchmark Suite for Complex Multimedia Applications. In: Proc. of the IEEE international workload characterization symposium. pp. 34-45, (2013).
18. Lee C., Potkonjak M., Mangione –Smith W.H.: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems. In: Proc. of the IEEE/ACM international symposium on Micro-architecture. pp. 330-335, (1997).
19. PrasannaSrinivasan V., Shanthi A.P.: A BBN-Based Framework for Design Space Pruning of Application Specific Instruction Processors, Journal of Circuits, Systems, and Computers. 25 (4), 1650028, (2016).

Dr. K.Balasubadra, received her B.E. Degree in Electronics and Communication Engineering in 1988 through PSNA College of Engineering and Technology, Dindigul under Madurai Kamaraj University and M.E Degree in Applied Electronics through the Government College of Technology, Coimbatore under Bharathiar University in 1997. She has obtained her Doctorate Degree in Information and

Communication Engineering from Anna University, Chennai, in 2009. She has 25 years of teaching experience to UG and PG classes and has guided many B.E. and M.E projects. Presently she is guiding ten PhD scholars and she is a research paper reviewer in conferences in National and International levels. Her research interests are Analog VLSI, Optical Communication and Wireless networks. She has published 5 papers in International Journals and 22 papers in conferences in National and International levels. She is a Life member of Indian Society for Technical Education (ISTE), member in Computer Society of India (CSI), Institute of Engineering and Technology (IET) and was a member in IEEE for more than 10 years. She is a recognized research supervisor of Anna University of Technology, Madurai.

Dr. A.P.Shanthi, received her M.E Degree in 1986. She has obtained her Doctorate Degree in Information and Communication Engineering from Anna University, Chennai, in 2006. She has more than 25 years of teaching experience to UG and PG classes and has guided many B.E. and M.E projects. Presently she is guiding five PhD scholars and she is a research paper reviewer in conferences in National and International levels. Her research interests are embedded system design, fault tolerant computing, computer architecture. She has published 17 papers in International Journals and 34 papers in conferences in National and International levels. She is a Life member of Indian Society for Technical Education (ISTE).

Dr.V.Prasanna Srinivasan, B.E, M.E, Ph.D, is an Associate Professor in Department of Information Technology, since December 2006. He obtained his B.E (CSE) from Madras University and M.E (Embedded Systems) from Anna University, Chennai, and Ph.D from Anna University, Chennai. He has been in the teaching profession for the past 13 years and has handled UG programs. His areas of interest include embedded systems, Design space exploration, fault tolerant systems. He has published 3 papers in referred International Journals. He is Life member of ISTE and CSI. He is serving as a reviewer for the Journal of Circuits, Systems and Computers.

International Journal of
NEW COMPUTER ARCHITECTURES AND THEIR APPLICATIONS

The *International Journal of New Computer Architectures and Their Applications* aims to provide a forum for scientists, engineers, and practitioners to present their latest research results, ideas, developments and applications in the field of computer architectures, information technology, and mobile technologies. The IJNCAA is published four times a year and accepts three types of papers as follows:

1. **Research papers**: that are presenting and discussing the latest, and the most profound research results in the scope of IJNCAA. Papers should describe new contributions in the scope of IJNCAA and support claims of novelty with citations to the relevant literature.
2. **Technical papers**: that are establishing meaningful forum between practitioners and researchers with useful solutions in various fields of digital security and forensics. It includes all kinds of practical applications, which covers principles, projects, missions, techniques, tools, methods, processes etc.
3. **Review papers**: that are critically analyzing past and current research trends in the field.

Manuscripts submitted to IJNCAA **should not be previously published or be under review** by any other publication. Plagiarism is a serious academic offense and will not be tolerated in any sort! Any case of plagiarism would lead to life-time abundance of all authors for publishing in any of our journals or conferences.

Original unpublished manuscripts are solicited in the following areas including but not limited to:

- Computer Architectures
- Parallel and Distributed Systems
- Storage Management
- Microprocessors and Microsystems
- Communications Management
- Reliability
- VLSI