ISBN 978-93-84422-37-0
2015 International Conference on Advances in Software, Control and Mechanical Engineering (ICSCME'2015)
Antalya (Turkey) Sept. 7-8, 2015 pp. 1-9

Using of Cuckoo Search Algorithm and Wave-Fronts Approach with Changing Angle for Tiled Iteration Space Scheduling of Two-Level Nested Loops

AliReza Hajieskandar[1], Javad Sohafi-Bonab[2], Simin Ghahramanian[3]

[1] Department of Computer Engineering Bonab Brranch, Islamic Azad University, Bonab, Iran;

[2] Department of Computer Engineering Bonab Brranch, Islamic Azad University, Bonab, Iran

[3] Sama Technical and Vocational Training College, Islamic Azad University, Bonab, Iran

Abstract: *Need for parallel processing is fueled by the existence of time consuming applications in different fields. The main rationale for parallel processing is that in most cases massive problems can be divided into smaller ones in order to be executed in parallel. In the optimization of sequential programs loops are the best candidates for optimization as they form up the major heavy computational part of any time consuming problem. Parallelization of application programs and duty timing is a very complex problem and decisively determines the efficiency of the target system. So the parallelization of nested loops is critical in accelerating programs' execution. For the conversion of nested loops into parallel loops, four steps must be taken orderly: the first I is the analysis of data dependency among the iterations of sequential loop iterations, stage II, is the tiling of loop iteration space, the III stage is, the automatic production of appropriate parallel code for tiled iteration space and the IV step, is the scheduling of tiled iteration space. This paper emphasizes on the fourth stage. As this problem is a polynomial - nondeterministic hard problem the use comprehensive search techniques to solve it is not suitable but apparently the use of Cuckoo Search Algorithm for solving this problem may be useful. In this paper by using the wave approach and its angle shift, a solution is presented in trying to find the optimal timing schedule. In this paper, by performing a variety of experimental tests and reviewing the practical results obtained from the proposed approaches and their comparison with the results of some former algorithms for this problem better outcomes have been achieved.*

Keywords: *Loop Parallelization, Scheduling, Wave-Front, Angular Wave, Cuckoo Search.*

1. Introduction

In past, high costs were spent on writing computational applications that included nested loops for being executed serially. If loops be executed in parallel the run time will drop drastically and also there will be no extra expenses for writing parallel programs. Therefore parallelization is a critical step in the rapid execution of programs. The parallelization consists of four stages of the analysis of data dependency, the tiling of iteration space, the generation of parallel code and the scheduling of tiled iteration space. The focus of this paper is over the fourth stage i.e. a NP-Hard Problem. As using deterministic approaches are not suitable for NP problems, the Cuckoo search algorithm was used for this purpose. The results obtained from the simulation of the proposed algorithm outperformed the results achieved by the former algorithms.

2. The Problem

The purpose from the scheduling of tiled iteration space of loops is outland considering the following example:

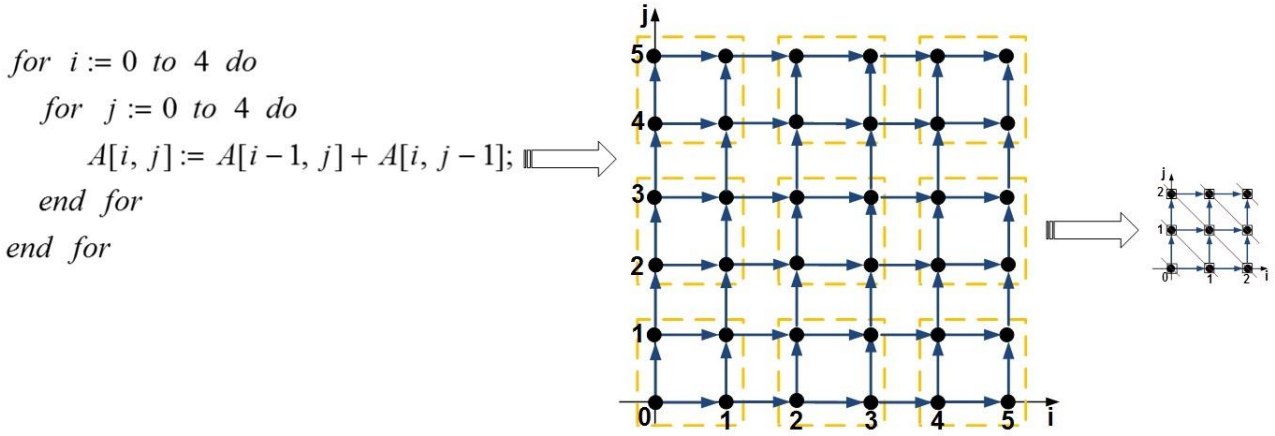

Fig. 1: An example of nested loop, it's iteration space and tiled iteration space [9]

The command inside the loops indicates that the iteration [i, j] depends on two iterations [i-1, j] and [i, j-1]. The iteration space is tiled as below (4 side by side iterations are considered as one tile).

As the above figure illustrates the mentioned inter-tile dependencies are also established. So before starting to execute the tile [i, j], the execution of tiles [i-1, j] and [i, j-1] should be finished. Since every tile runs over one processor the tiling of iteration space for loops reduces the communication volume among processors.

The purpose of scheduling is the allocation of tiles in such a way that minimizes the makespan of all tiles. To achieve this goal, three costs are considered. The execution cost of a tile over a processor Vcomp[1], which is the same for all tiles, the cost of communication and message sending between the existing tiles and prerequisite tiles regarding the first dimension, Vcomm[2] (1) and the same cost for the second dimension Vcomm (2). These costs are taken as the problem inputs and they exist if the J^S tile and its prerequisite tiles run over different processors. If the processors be same, the communicational cost will be zero. According to the direct connection of each processor to the other ones the communicational cost of all processors is the same. So we can extract the makespan of the tile J^S via the following relation:

$$completion\,time(J^S) = Max\begin{pmatrix} completion\,time(J'^S) + V_{comm}(1), \\ completion\,time(J''^S) + V_{comm}(2), \end{pmatrix} \quad (1)$$

One of the algorithm's output parameters is the scheduling makespan which stands for the total time required for the execution of all existing tiles. This parameter is calculated using the following relation.

Tiled iteration space

$$\text{makespan} = \max(\text{completion time}(J^S)) \quad , J^s \in \quad (2)$$

3. An Overview of Cuckoo Search Algorithm

[1] volume of computation
[2] volume of communication

Cuckoo search is a relatively recent nature-inspired metaheuristic algorithm, developed by Xin-She Yang and Suash Deb in 2009. CS was inspired by the brood parasitism of some cuckoo bird species, in combination with the Lévy flights [17–19] random walks. Cuckoos are catching scientists' interest because of their aggressive reproduction strategy. Some species lay their eggs in communal nests of other host birds (often other species), and may remove others' eggs to increase the hatching probability of their own eggs.

CS idealized rules can be summarized as:

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest.
- The best nests with high quality of eggs will carry over to the next generations.
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0,1]$ where the host bird can either throw the egg away, or abandon the nest and build a completely new nest [28].

These rules are used to draw the basic cuckoo search algorithm in Figure 2.

1: Objective function f (x), x = (x1, . . . , xd)^T
2: Generate initial population of n host nests xi(i = 1, . . . , n)
3: **while** (t <MaxGeneration) or (stop criterion) **do**
4: Get a cuckoo randomly by L´evy flights
5: Evaluate its quality/fitness Fi
6: Choose a nest among n (say, j) randomly
7: **if** (Fi > Fj) **then**
8: replace j by the new solution;
9: **end if**
10: A fraction (pa) of worse nests are abandoned and new ones are built;
11: Keep the best solutions (or nests with quality solutions);
12: Rank the solutions and find the current best
13: **end while**
14: Postprocess results and visualization

Fig. 2: Pseudo code of Cuckoo search algorithm [17]

While exploring new solutions, it is necessary to control the Lévy flights random walks, to avoid large moves, causing the solutions to jump outside of the search space. A step size factor that is defined according to the scale of the problem of interest should be used for this purpose. This might be an interesting subject for more research, studying the optimal utilization of the Lévy flight in optimization; for simplicity, a typical step size factor of 0.01, as suggested by the authors, is being used in this study [17, 28 and 29]..

4. Related Works

Many approaches have been offered to schedule the iteration space of two level nested loops. Any of these methods somehow tries to decrees the run time of repeated loops in a parallel manner [1, 5, 7, 11, 15, 16, 18, 20-24, 26 and 27]. In some methods, in order to minimize the overall execution time of iterations, the communication time of processors overlaps with the time of internal calculations of processors [3, 7 and 24]. Also some other approaches inspired by the wave-front method, eliminate the need for communicating of many processors and assign dependent caches to the same processor [7, 9, 15, 24]. In other approaches, the critical path for the task graph resulted from the loop iterations' space is calculated and in order to obtain load balancing, other non-critical iterations are distributed among processors according to their data dependency [1].

We in the former articles [9 and 10] using the wave-front method introduced two genetic-based algorithms for scheduling two level nested loop. One of the methods with angle shift and the other without it can culminate at the proper scheduling of the problem in question.

We also achieved optimum results in [11 and 12], by shifting the wave angle in the following four approaches: the horizontal block, the vertical block, the horizontal cyclic and the vertical cyclic. In this paper, we extended this approach with genetic algorithm combination for three-level space.

We also achieved optimum results in [8], by combination of genetic algorithm and changing wave-fonts angle approach for tiled iteration space scheduling of three-level nested loops in the following six approaches:

the horizontal block, the vertical block, the depth block, the horizontal cyclic, the vertical cyclic and the depth cyclic.

Andronikos and et. al. in [1] suggested a multinomial algorithm for efficient scheduling of uniform dependency loops by outlining precise criteria for a number of engaged processors in which they identified upper and lower limits for optimum number of required processors which minimizes the total execution time and then performs a binary search between the two borders.

In [15 and 16], the four approaches of horizontal block, vertical block, and horizontal cyclic and vertical cyclic are presented. These approaches, take several neighbouring tiles as one block and hence they remove enormous inter- processor contacts and allocate dependent tiles to a single processor appropriately.

Moldovan, Shang and Darte et al benefited the hyper-plane method to explore an optimal linear schedule by using Diophantine equations [16], linear programming in subspaces [25] and accurate programming [4]. Hence more computations will be rendered in parallel.

Most of approaches which concentrated on optimum scheduling originate from hyper-plane methods [13]. The main objective of these methods is to classify computations into well-defined groups called wave or hyper-plane by using linear transformations .All iterations existing inside a loop can be executed simultaneously. Darte in [4] showed the optimality of this method.

Athanasaki and et. al. in [2] introduced the following four methods of the cyclic assignment schedule, the mirror assignment schedule, the cluster assignment schedule and the retiling schedule for scheduling of tiled iteration space to clustered system with a given number of SMP(Symmetric Multi-Processors) nodes and applied two execution schemes including the overlapping execution scheme and non-overlapping execution scheme.

Doritos and et. al. in [6] introduced a low-cost algorithm which using geometrical calculations generates the typical schemes consisting of iteration subsets which can be executed earlier in a certain k-time phase.

5. Proposed Strategy

The allocation of tiles for being executed over processors is performed according to the wave approach. For this the tiles without inter-dependencies which can be run over different processors are considered as one wave. These waves which should be executed in order can form various angles with the coordinates of tiled space which are shown by the "a" and "b" angles. For example if "a" and "b" equate to “1” then the angle will be 45 degrees. By shifting this angle the existent tiles over the waves get closer to each other. The Figure 3 shows the wave along with its angle.

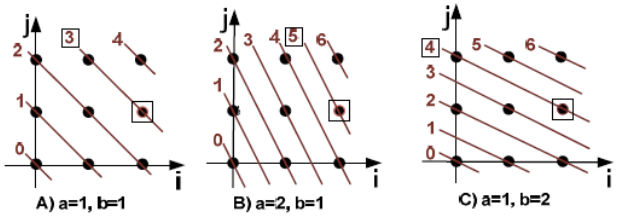

Fig. 3: An example of wave-front approach and it’s Angle [9]

As shown in the above figure - part a, there are one tile on the first and last waves, two tiles on the second and penultimate waves, and maximum number of tiles on the middle wave. But part b shows that the number of existing tiles on each wave is almost equal. So according to the fact that each tile is executed on one processor, by changing the waves’ angle , the load balance among processors improves with implications for their better efficiency .In continue the proposed algorithm is reviewed.

The main idea of our work is that improved CS seeks good solutions found using local search in areas specified by L´evy flights. CS and its inspiration source are structured around five main elements: egg, nest, objective function, search space and L´evy flights. These key elements can have important meanings for combinatorial problems […].

5.1. The Egg

If we consider that a cuckoo lays a single egg in one nest, we can gives eggs the following properties:

• An egg in a nest is a solution adopted by one individual in the population;

• An egg of the cuckoos is a new solution candidate for a place in the population.

Each egg is equivalent to a processor allocation matrix for tiled iteration space and coefficients vector witch that show the angle of wave-fronts. Coefficients vector is binary encoded. A sample of an egg is shown in figure 4[17].

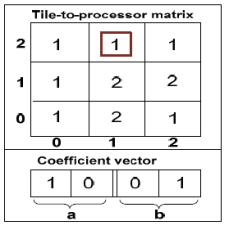

Fig .4: An example of an egg [9]

5.2. The Nest

In CS, the following features can be imposed concerning a nest:

• The number of nests is fixed;

• A nest is an individual of the population and the number of nests equal to the size of the population;

• An abandoned nest involves the replacement of an individual of the population by a new one;

In this problem, a nest is shown as an individual in the population with its own assign matrix and coefficient vector. Obviously, a nest can have multiple eggs for future extensions. In the present paper, each nest only contains a single egg, for simplicity [17].

5.3. The Objective Function

Each solution in the search space is associated with a numeric objective value. So the quality of a solution is proportional to the value of the objective function. In CS, a nest egg of better quality will lead to new generations. This means that the quality of a cuckoo's egg is directly related to its ability to give a new cuckoo.

The fitness function of this problem is shown in figure 5.

Scheduling Completion Time = 0;
foreach wavefront no., *wn*, $lwn \le wn \le hwn$ **do**
foreach Processor *Pi*, $0 \le i \le N-1$ **do** ProcessingTime (P_i)= 0;
foreach tile $J^S=(j_1^S, j_2^S)$ on wavefront *wn* such that $a.j_1^S + b.j_2^S = wn$ **do**
Find processor no., P_i, assigned to J^S ; Assume $J'^S = (j_1^S - 1, j_2^S)$ and $J''^S = (j_1^S, j_2^S - 1)$;
Find processor no., $P_{i'}$ and $P_{i''}$ assigned to J'^S and J''^S, respectively;
ProcessingTime $(P_i) += V_{comp}.T_{comp}$: computation time of the tile;
(1) **if** J^S is only dependent on J'^S **then**
if $P_i \neq P_{i'}$ **then** ProcessingTime $(P_i) += V_{comm}(1).T_{comm}$: communication time 1
(2) **else if** J^S is only dependent on J''^S **then**
if $P_i \neq P_{i''}$ **then** ProcessingTime $(P_i) += V_{comm}(2).T_{comm}$: communication time 2
(3) **else if** J^S is dependent on both J'^S and J''^S **then**
if $P_{i'} \neq P_{i''}$ **then**
if $P_i \neq P_{i'}, P_{i''}$ **then** ProcessingTime $(P_i) += Max(V_{comm}(1).T_{comm}, V_{comm}(2).T_{comm})$
else if $P_i \neq P_{i'}, P_{i''}$ **then** ProcessingTime $(P_i) += V_{comm}(1).T_{comm} + V_{comm}(2).T_{comm}$
ProcessingTime (*wn*) = Max (ProcessingTime (P_i)) : $\forall P_i \in [0..N-1]$;
SchedulingCompletionTime+=ProcessingTime (*wn*);

Fig. 5: The fitness function [9]

5.4. The Search Space

The search space is array of records. Each record has 3 fields; Assign matrix, coefficient vector and fitness value.

5.5. *L´evy flights*

L´evy flights have as a characteristic of an intensified search around a solution, followed by big steps in the long run. in some optimisation problems, the search for a new best solution is more efficient via L´evy flights. In order to improve the quality of search, we will associate the step length to the value generated by L´evy flights as outlined in the standard CS [17].

6. Experimental Results

In order to simulate the results of proposed algorithm, CSALS3, an application is implemented in C# 2010 environment for this algorithm, and BlockV, BlockH, CyclicV, CyclicH and BCGALS algorithms .Various tests were implemented by using them and the results of these tests have been shown in the following.

6.1. Reliability of Algorithm

To investigate the error rate of the proposed algorithm, a problem of 4 × 8 in dimensions with parameters (UB1=3, UB2=7, NP=2, Vcomp=2, Vcom1=10, Vcom2=10) is considered. The algorithm was executed 10 times and obtained solutions are shown in figure 6.

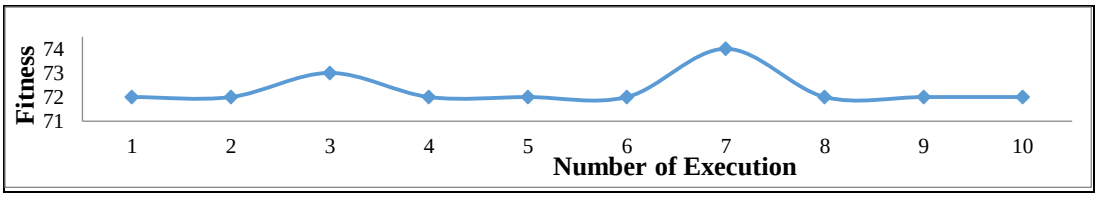

Fig. 6: Reliability of proposed algorithm for 4*8 problem

6.2. Sustainability of Algorithm

According to tests, the results of the algorithm are converging.

6.3. Quality of produced results

According to our related approaches [8-11], we run the simulated software for various experiment and select some of them here.

TABLE I: The outcome results from implementing the proposed algorithm and former algorithms

Parameters	Algorithm & Result					
	BlockV	BlockH	CyclicV	CyclicH	BCGALS	CSALS
UB1=5, UB2=5, NP=3, Vcomp=9, Vcom1=7, Vcom2=7	229	229	242	242	219	217
UB1=3, UB2=5, NP=2, Vcomp=9, Vcom1=7, Vcom2=7	179	201	210	210	177	175
UB1=3, UB2=5, NP=3, Vcomp=9, Vcom1=7, Vcom2=7	164	179	164	178	153.6	152.3
UB1=3, UB2=7, NP=2, Vcomp=2, Vcom1=10, Vcom2=10	134	110	228	258	80	72
UB1=9, UB2=4, NP=3, Vcomp=2, Vcom1=10, Vcom2=10	150	170	244	244	162.4	120
UB1=7, UB2=5, NP=3, Vcomp=2, Vcom1=10, Vcom2=10	150	146	220	220	174	112

[3] Cuckoo Search Algorithm for Loop Scheduling

7. Conclusion and Future Work

In this paper, taking the benefits of angle shifts, an algorithm was proposed which due to its potential in improving the loading balance among processors and consequently optimizing of their usage, minimized the execution time of tiles. As he results show, the proposed algorithm (CSALS) yielded better results in 95% of cases relative to the Block, Cyclic and Genetic methods.

Looks can be combined with meta-heuristic algorithms achieved better results.

8. Acknowledgements

AliReza Hajieskandar received the B.Sc. in Software Engineering from Islamic Azad University shabestar Branch, Iran and the M.Sc. degree in Software Engineering from Islamic Azad University Qazvin Branch, Iran. He is preceptor of Software Engineering at the Islamic Azad University Bonab Branch. His research interests include compilers, super-compilers, parallel processing, evolutionary computing and algorithms.

Javad Sohafi Bonab received the B.Sc. in Software Engineering from Bonab PNU and the M.Sc. degree in Software Engineering from Islamic Azad University Qazvin Branch, Iran. He is preceptor of Software Engineering at the Islamic Azad University Bonab Branch. His research interests include compilers, super-compilers, parallel processing, evolutionary computing and algorithms.

Simin Ghahramanian received the B.Sc. in Software Engineering from Islamic Azad University shabestar Branch, Iran, and the M.Sc. degree in computer system architecture. she is preceptor of Software Engineering at the Sama Technical and Vocational Training College, Islamic Azad University, Bonab Branch. His research interests include parallel processing, Network Computing and NOC Testing

9. References

[1] T. Andronikos, M. Kalathas, F. M. Ciorba, P. Theodoropoulos, and G. Papakonstantinou, "An Efficient Scheduling of Uniform Dependence Loops", Computing Systems Laboratory, Department of Electrical and Computer Engineering, National Technical University of Athens, Zographou Campus, Athens, Greece, pp. 1-10, 2004.

[2] M. Athanasaki, E. Koukis, and N. Koziris, "Scheduling of Tiled Nested Loops onto a Cluster with a Fixed Number of SMP Nodes", 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, IEEE, 2004.

http://dx.doi.org/10.1109/empdp.2004.1271475

[3] M. Athanasaki, A. Sotiropoulos, G. Tsoukalas, N. Koziris, and P. Tsanakas, "Hyperplane Grouping and Pipelined Schedules: How to Execute Tiled Loops Fast on Clusters of SMPs", Journal of Supercomputing, vol. 32, pp. 197-226, 2005.

http://dx.doi.org/10.1007/s11227-005-0298-8

[4] A. Darte, L. Khachiyan, and Y. Robert, "Linear Scheduling Is Nearly Optimal", pp. 73-81, 1991.

[5] A. Darte, Y. Robert, and F. Vivien, "Scheduling and Automatic Parallelization", 2000.

[6] I. Drositis, T. Andronikos, A. Kokorogiannis, G. Papakonstantinou, and N. Koziris, "Geometric Pattern Prediction and Scheduling of Uniform Dependence Loops", 2001.

[7] G. Goumas, A. Sotiropoulos, and N. Koziris, "Minimizing Completion Time for Loop Tiling with Computation and Communication Overlapping", IEEE, pp. 1-10, 2001.

http://dx.doi.org/10.1109/ipdps.2001.924976

[8] Hajieskandar and R. Etemadi, "combination of genetic algorithm and changing wave-fonts angle approach for tiled iteration space scheduling of three-level nested loops ", (ICAEME'2015) Jan. 7-8, 2015 Dubai (UAE)

[9] A. Hajieskandar and S. Lotfi, "Parallel Loop Scheduling Using an Evolutionary Algorithm", pp. 314-319, August 2010.

[10] A. Hajieskandar and S. Lotfi, "Using an Evolutionary Algorithm for Scheduling of Two-Level Nested Loops", pp. 100-105, May 2011.

[11] A. Hajieskandar, S. Lotfi, and S.Ghahramanian, "Two Level Nested Loops Tiled Iteration Space Scheduling By Changing Wave-Front Angles Approach", International Journal of Advanced Research in Computer and Communication Engineering, vol. 1, Issue 3, pp. 126-133, May 2012.

[12] A. Hajieskandar, S. Lotfi, and S.Ghahramanian, "Use of Changing Wave-Front Angles Approach for Tiled Iteration Space Scheduling of Three-Level Nested Loops", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 2, February 2014.

[13] L. Lamport, "The Parallel Execution of Do Loops", Comm. of the ACM, vol. 37, No. 2, pp. 83-93, February 1974.

http://dx.doi.org/10.1145/360827.360844

[14] S. Lotfi and S. Parsa, "Parallel Loop Generation and Scheduling", Journal of Supercomputing, vol. 50, No. 3 pp. 289-306, 2009

http://dx.doi.org/10.1007/s11227-008-0262-5.

[15] N. Manjikian and T. S. Abdelrahman, "Scheduling of Wavefront Parallelism on Scalable Shared-Memory Multiprocessors", pp. 1-10, 1996.

http://dx.doi.org/10.1109/icpp.1996.538567

[16] D. I. Moldovan and J. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays", vol. C-35, No. 1, pp. 1-11, 1986.

[17] A. Ouaarab, B. Ahiod and X. Yang, "Discrete Cuckoo Search Algorithm for the Traveling", NEURAL COMPUTING AND APPLICATIONS, Vol. 24, No. 7-8, pp. 1659-1669, JUNE 2014.

[18] T. W. O'Neil, "Techniques for Optimizing Loop Scheduling", Ph. D. Thesis, The Graduate School of the University of Notre Dame, Indiana, 2002.

[19] S. Parsa and S. Lotfi, "A New Approach to Parallelization of Serial Nested Loops Using Genetic Algorithms", Journal of Supercomputing, vol. 36, No. 1, pp. 83–94, 2006

http://dx.doi.org/10.1007/s11227-006-3557-4.

[20] S. Parsa and S. Lotfi, "A New Genetic Algorithm for Loop Tiling", Journal of Supercomputing, vol. 37, No. 3, pp. 249–269, 2006.

http://dx.doi.org/10.1007/s11227-006-6367-9

[21] S. Parsa and S. Lotfi, "Code Generation and Scheduling for Parallelization of Multi-Dimensional Perfectly Nested Loops", pp. 20-22, February 2007.

[22] S. Parsa and S. Lotfi, "Wave-Fronts Parallelization and Scheduling", pp. 382-386, November 18-20, 2007

http://dx.doi.org/10.1109/iit.2007.4430369.

[23] D. L. Pean, H. T. Chua, and C. Chen, "A Release Combined Scheduling Scheme for Non-Uniform Dependence Loops", Journal of Information Science and Engineering, vol. 18, pp. 223-255, 2002.

[24] F. Rastello and Y. Robert, "Automatic Partitioning of Parallel Loops with Parallelepiped-Shaped Tiles", vol. 13, No. 5, pp. 460-470, May 2002.

[25] W. Shang and J. A. B. Fortes, "Time Optimal Linear Schedules for Algorithms with Uniform Dependencies", IEEE Transactions on Computers, vol. 40, No. 6, pp. 723-742, 1991.

http://dx.doi.org/10.1109/12.90251

[26] J. Xue, "On Tiling as a Loop Transformation", vol. 7, No. 4, pp. 409-424, 1997.

[27] C. T. Yang and K. Cheng, "An Enhanced Parallel Loop Self-Scheduling Scheme for Cluster Environments", Journal of Supercomputing, vol. 34, pp. 315-335, 2005

http://dx.doi.org/10.1007/s11227-005-0787-9.

[28] X. Yang and S. Deb, "Cuckoo Search via L´evy Flights", 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), IEEE, pp. 210-214, 2009.

http://dx.doi.org/10.1109/NABIC.2009.5393690

[29]H.Zheng and Y.Zhou, "A Novel Cuckoo Search Optimization Algorithm Base on Gauss Distribution", Journal of Computational Information Systems, pp. 4193–4200 (2012).