Dirasat, Pure Sciences, Volume **35**, No. **2**, 2008

A Heuristic Genetic Algorithm for Independent Task Scheduling

Isra N. ALKallak and Ruqaya Z. Sha'ban **

ABSTRACT

This research tackles genetic algorithm techniques in scheduling independent tasks for multiprocessor by proposed heuristic genetic algorithm resulting from combining two classical list heuristics of scheduling independent tasks, in order to gaining an optimal scheduling with the least period of time (schedule length). The researcher proved is more efficient than using a separate algorithm of the two list scheduling algorithms.

Keywords: Genetic Algorithm, Task Scheduling.

1. INTRODUCTION

A parallel program can be viewed as a collection of tasks, which execute on a set of processors. Tasks can execute in sequence or at the same time on two or more processors. The objective of scheduling is to find an assignment of tasks to processors and execution order so that program time is minimized.

There are many important scheduling problems in the computer sciences. One important class of scheduling heuristics is list scheduling which considers all tasks in an order given by a list, and selects the best processor for each of them according to some criterion. We assume that the tasks have a given processor allocation.

The problem of non preemptively scheduling a set of tasks with arbitrary computation times on multiprocessors in known to be NP-complete. The problem here is complicated since the number of processors may be larger than one and the tasks are independent (Nossal, 2000).

Task scheduling can be classified as static and dynamic. In static algorithms, the assignment of tasks to processors and time at which tasks start execution are determined a priori (Mahmood, 2000).

The research tackles static scheduling allows only one process per processor, multiprocessor scheduling refers to the process in which tasks in a given task graph are assigned to processing elements, and the time in which the tasks are executed are determined. The tasks are independent structure. The processing times of the tasks can take arbitrary values .

The algorithm presented in this research is based on the heuristic genetic algorithm technique is a tool to find an assignment of tasks to processors, and an execution order so that, program execution time is minimized (schedule length). The schedule length provide a measure of processor utilization .

This research is organized as follows: scheduling model, task graph and Gantt chart, independent tasks and basic list scheduling in section 2. section 3 presents a genetic algorithm, genetic algorithm procedure and cycle of a standard genetic algorithm. The section 4 contains the proposed genetic list scheduling algorithm, experimental evaluation and result. Also the section 4 include conclusion.

1.1. Related Work

The problem of scheduling has been studied and a number of algorithms has been proposed and many heuristics have been developed, each of which may find optimal or near optimal scheduling. Grajcar (1999) proved that the combination of genetic, and list scheduling to be powerful. Esquivel et al. (2000) showed that the problem of all allocating a number of tasks in a multiprocessor and tasks are non-preemptive with a genetic approach that provide good solutions. The schedule which was proposed by Nossal (2000) is based on genetic algorithms. Auyeung et al. (2003) proposed a genetic algorithm that find a good combination of four list heuristics to produce a schedule with shortest execution time.

* Department of Medical Basic Sciences, College of Nursing; and Computer Unit., College of Medicine, University of Mosul, Iraq. Received on 25/11/2007 and Accepted for Publication on 8/7/2008.

2. Scheduling Model

A task graph is simplified representation of a parallel program execution. It will be assumed that the execution time for each task is known a prior. The execution time of each task can either be interpreted as maximum processing time. In this case the scheduling length is the maximum time to complete the graph. Or as the expected processing time of the run times considered as random variables (Hwang and Briggs, 1984); (Muntz, 1969); (Ramamoorthy et al., 1972). A schedule is an allocation of tasks to processors which can be represented by a Gantt chart. In a Gantt chart, the initiation and ending times for each task in the available processors is indicated, and the makespan (total execution time of the parallel program) of the schedule can be easily derived (Esquivel et al., 2000).

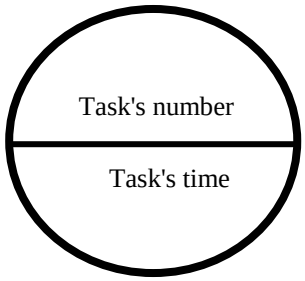

Figure (1) Task graph

2.1. Task Graph and Gantt Chart

A task graph is a graph in which each node represents a task to be performed. Each node generally has the form shown in Figure (1). The Gantt chart gives the schedule shows the start and finish times for all tasks (Yun, 1997). When one or more processors remained idle period that means no task allocated to them, which the processors had a symbol ϕ in Gantt chart (Thiebaut, 1995).

2.2. Independent Tasks

The system is a set of {M} identical processors $\{p_1,----,p_M\}$ that operate in parallel. The set $T= \{T_1,----T_n\}$ of tasks to be executed consists of independent tasks (no ordering constraints between them), each of which has an execution time T_i and requires only one processor. The only processing constraints are that no processor can execute more than one task at a time and that, once a processor begins executing a task T_i, it continues executing T_i until its completion T_i time units later (non-preemptive scheduling) (Garey et al., 1978). It is possible for all processes in a graph to be independent of each other. In this case there is no precedence relation or partial ordering between the processes and all the processes can be scheduled concurrently (Hwang and Briggs, 1984).

2.3. Basic List Scheduling

A list scheduling is the method of scheduling static task graph. In list scheduling, each task graph node is assigned a priority, and a list of nodes is constructed in order of decreasing priority. Whenever a processor is available, a ready node with the highest priority is selected from the list, and assigned to the processor. If more than one node has the same priority, a node is selected randomly. List schedulers differ only in how they assign priorities, but this result in different schedules because node are selected in different order. A list schedule keep a list of tasks that are ready to be scheduled. Figure (2) shows the general algorithm list scheduling (Arroyo, 1997); (Auyeung et al., 2003).

```
Begin
    Assign priorities to tasks.
Repeat
    Select a task.
    Select a processor to run the task.
    Assign the task to the processor.
Until all tasks are scheduled.
End.
```

Figure (2)

One important class of scheduling heuristics list scheduling which considers all tasks an order given by a list, and selects the best processor for each of them according to some criterion (Grajcar, 1999). The assignment of priorities to tasks results in different schedules because tasks are selected for execution in different orders. The proposed solution is to use a genetic algorithm to find combination of the two algorithms. First, that longest processing time (LPT) (Gonzalez, 1977). Second, that shortest processing time (SPT) (Gonzalez, 1977).

3. Genetic Algorithm

A genetic algorithm is a heuristic method used to solve optimization problems in many fields. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and

crossover. Genetic algorithms are typically implemented as a computer simulation, in which a population of representations (called chromosomes) of solutions (called individuals) to an optimization problem evolves to ward better solution (http://en.wikipedia.org /wiki /genetic algorithm). A problem to be solved by a genetic algorithm to be encoded in a way, because algorithms act bit strings. Each bit string encodes one part of the information on the problem solution and is called a gen. Genes are grouped to chromosomes, one or more chromosomes form an individual. Each individual encodes a complete solution to the given problem. A certain number of individuals form a population (Nossal, 2000). A chromosome is a representation in which there is a list of elements called genes, and an allele is one member of a pair or series of genes that occupy a specific position on a specific chromosome, see figure (3) (Mahmood, 2000).

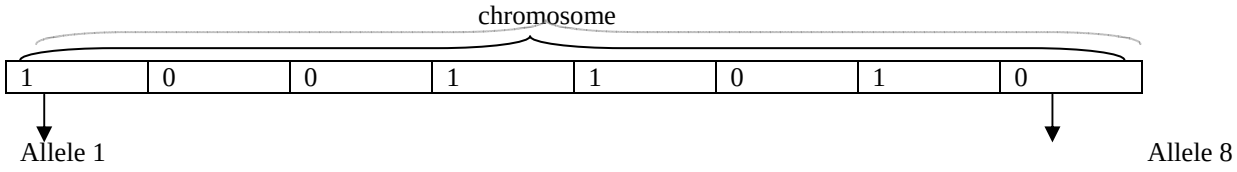

Figure (3) An 8-bit chromosome

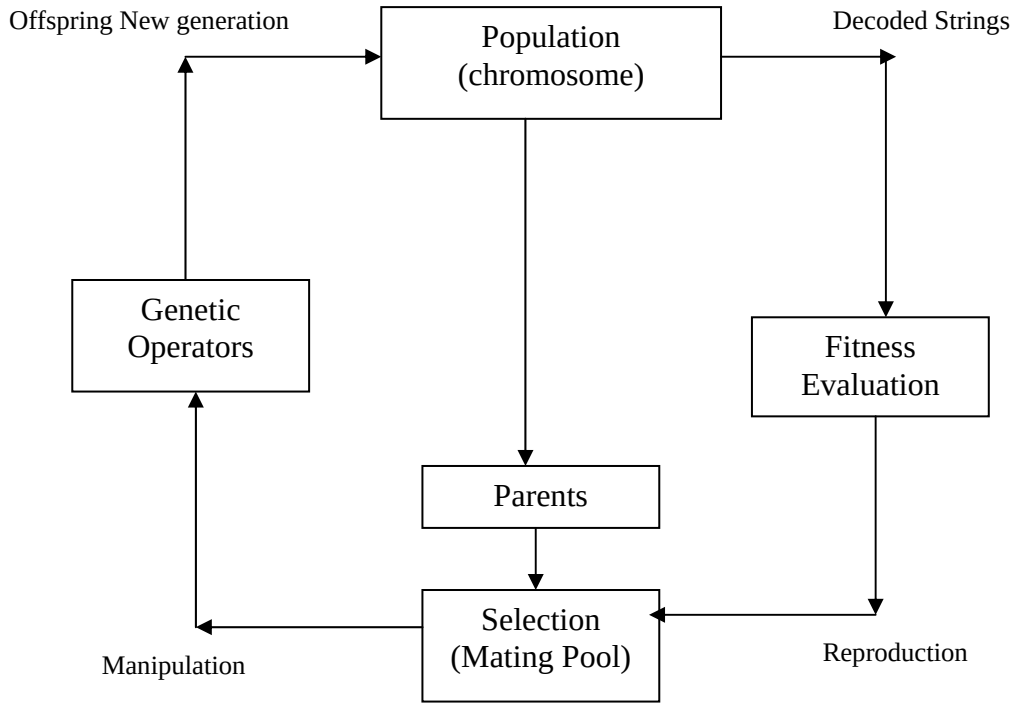

Figure (4) The cycle of a genetic algorithm

3.1. Genetic Algorithm Procedure

A typical genetic algorithm requires things to be defined

1- Initialization: Initially many individual solutions are randomly generated to form an initial population size depends on the nature of the problem, covering possible solutions (the search space) (http://en.wikipedia.org /wiki /genetic algorithm).

2- Selection: During each successive epoch, a proportion of the existing population is selected generation. Individual solutions are selected through a fitness-based process. The fitness of each solution and preferentially select the best solutions (http://en.wikipedia.org /wiki /genetic algorithm).

In this research used the selection called "selection rank with elitism". "Elitism" first introduced by Kenneth

De Jong (1975), that forces the genetic algorithm to retain in some number of the best individuals at each generation. Such individuals can be lost if they are not selected to reproduce or if they are destroyed by crossover or mutation (Mitchell, 1998).

1- Let *N* population size .
2- Generate random number of chromosomes.
3- For each task *t* in the tasks graph compute the following:
P(t)=(lpt*wlpt)+(spt*wspt)
{ lpt is the priority assigned to *t* in LPT algorithm; spt is the priority that assigned to *t* in SPT algorithm; wlpt & wspt is the weight assigned to LPT& SPT algorithms respectively}.
4- Sort tasks in decreasing order of fitness.
5- Find the selection using "selection rank with elitism" by:
   - 20 highest fitness "elite" rules were copied to the next generation without modification, {let assume *E*}.
   - The remaining 80 rules for the next generation were formed by single point crossovers between randomly chosen pairs of elite rules, {let assume *D=N-E*}.
   - The offspring from each crossover were mutated to the next generation by single point crossovers between randomly chosen pairs of elite rules.
   - If termination condition has been terminating then go to step 6 Else generational process is repeated by go to step 3 .
6- stop

Figure (5)

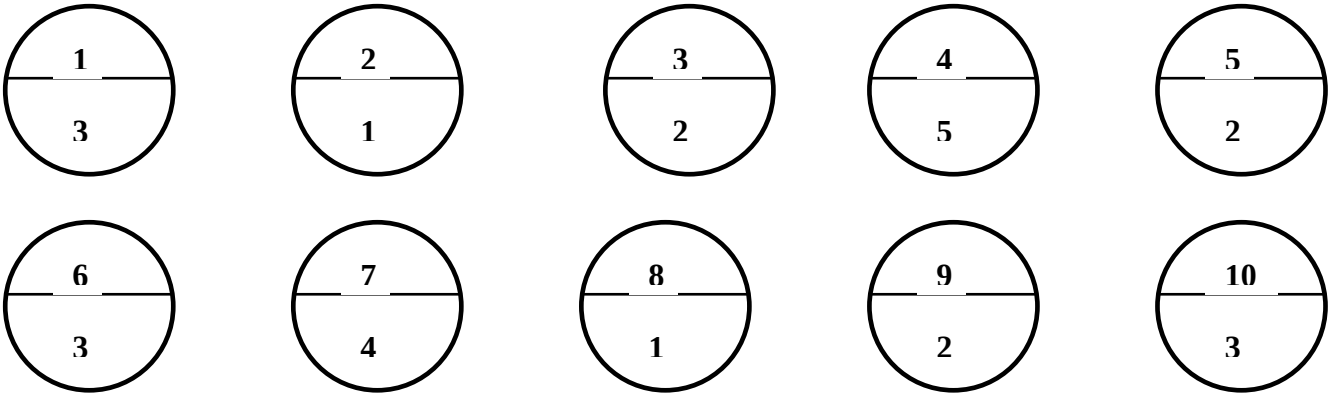

Figure (6) Task graph

The population was ranked by fitness and the 20 highest fitness "elite" rules were copied to the next generation without modification. The remaining 80 rules for the next generation were formed by single point crossovers between randomly chosen pairs of elite rules. The offspring from each crossover were mutated (Mitchell, 1998).

3- Reproduction: The next step is to generate a second generation population of solutions from the genetic operators. Crossover (also called recombination). For each new solution to be produced, a pair of "parent" solutions is selected from selected previously. By producing a "offspring" solution. The process continues until a new population of size is generated (http://en.wikipedia.org /wiki/genetic algorithm).
4- Termination: This generational process is repeated until a termination condition has been terminating (http:// en.wikipedia.org /wiki/genetic algorithm).

Figure (7) Gantt chart for LPT algorithm

Figure (8) Gantt chart for SPT algorithm

3.2. Standard Genetic Algorithm

A genetic algorithm operates through a simple cycle of stages:

i. Creation of a "population" of strings.
ii. Evaluation of each string.
iii. Selection of best strings.
iv. Genetic manipulation to create new population of string (Konar, 2000).

The cycle of a genetic algorithm in figure (4):

4. The Proposed Genetic List Scheduling Algorithm

Figure (5) shows the Proposed genetic list scheduling algorithm in the research:

4.1. Experimental Evaluation and Results

The experiments implemented proposed genetic algorithm and for evaluation of the algorithm, the Figure (6) was task graph composed of (10) task, and (1-5) unit of execute time, randomized initial population of size to (16) individuals, one point crossover and bit inversion mutation.

The maximum number of generations was fixed to (4), probabilities for crossover and mutation were fixed to 0.2, the genetic algorithm uses a one byte (8bit) to represent the chromosome, many runs were performed, stop criterion was used like classical measures to accept the solution. We wrote programs in C^{++} and visual basic language, Gantt chart generated by the scheduled for one

and two processors. The researchers executed LPT, SPT and proposed genetic algorithms, the Gantt chart generated by the scheduled for one and two processors in the figures (7), (8) and (9), respectively.

Figure (9) Gantt chart for proposed genetic algorithm

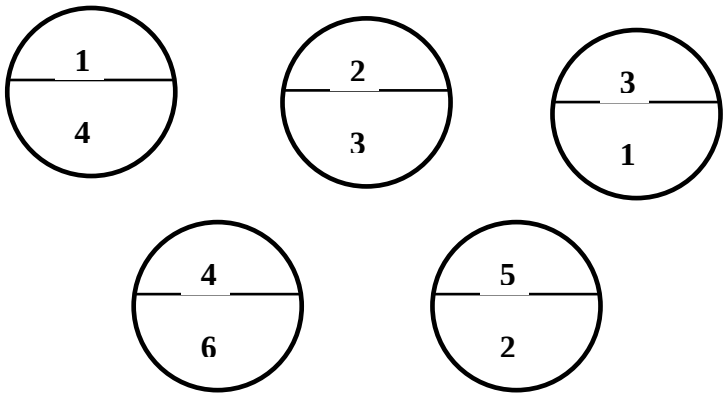

Figure (10) Task graph

Another example for implementation for proposed genetic algorithm in the figure (10) was task graph composed of (5) task, and (1-6) unit of execute time, randomized initial population of size to (10) individuals. The maximum number of generations was fixed to (3). Below the Gantt chart generated by the scheduled for one and two processors for LPT, SPT and proposed genetic algorithms in the figures (11), (12), and (13), respectively.

5. CONCLUSION

In this research we have the heuristic genetic algorithm for static list scheduling find the optimal scheduling by the combination of genetic and list scheduling algorithms proved to be very powerful. The algorithm outperformed them. It shows a better approach to the problem. We have found that elitism significantly improves the genetic algorithm's performance. In this research we approach the heuristic technique genetic

algorithm to minimize the schedule length. After many runs were performed of the proposed genetic algorithm in this research to be noticed the genetic algorithm is very powerful.

Figure (11) Gantt chart for LPT algorithm

Figure (12) Gantt chart for SPT algorithm

Figure (13) Gantt chart for proposed genetic algorithm

Appendix for figure (6)

Order of population	The chromosome of population: Weight Long Processing Time (wlpt)	The chromosome of population: Weight Short Processing Time (wspt)	Task sort in order of fitness	The chromosome of population was ranked in order of fitness	The parent in population was selected by "elite"		Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Long Processing Time (wlpt)	Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Short Processing Time (wspt)
0	0	7	3 6 9 5 0 8 4 2 7 1	4	4	20% are copied to the next generation without modification	12	14
1	0	12	3 6 9 5 0 8 4 2 7 1	12	12		9	15
2	7	13	9 3 6 8 5 4 0 7 2 1	7	7		9	13
3	5	11	3 9 6 5 8 0 4 7 2 1	6			0	14
4	12	14	9 8 3 6 7 5 4 1 0 2	11			12	14
5	8	4	7 8 1 9 4 5 2 6 3 0	2			9	13
6	14	8	7 8 9 1 4 5 2 6 3 0	3			1	13
7	9	13	9 3 8 6 5 7 4 0 1 2	15		80% the parents are chosen from 'elite ' by using crossover and mutation operation	9	13
8	3	7	3 9 6 5 8 0 4 7 2 1	14			8	5
9	11	0	7 1 8 4 2 9 5 0 6 3	10			11	15
10	0	12	3 6 9 5 0 8 4 2 7 1	5			5	7
11	5	15	3 9 6 5 8 0 4 7 2 1	1			4	6
12	9	15	9 3 6 8 5 4 7 0 2 1	13			6	14
13	7	4	7 8 9 1 4 5 2 6 3 0	9			13	15
14	0	12	3 6 9 5 0 8 4 2 7 1	8			9	15
15	5	8	9 3 6 8 5 4 7 0 2 1	0			8	11

Generation no. 1

Order of population	The chromosome of population: Weight Long Processing Time (wlpt)	The chromosome of population: Weight Short Processing Time (wspt)	Task sort in order of fitness	The chromosome of population was ranked in order of fitness	The parent in population was selected by "elite"		Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Long Processing Time (wlpt)	Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Short Processing Time (wspt)
0	12	14	9 8 3 6 7 5 4 1 0 2	13	13	20% are copied to the next generation without modification	13	5
1	9	15	9 3 6 8 5 4 7 0 2 1	9	9		11	15
2	9	13	9 3 8 6 5 7 4 0 1 2	4	4		12	14
3	0	14	3 6 9 5 0 8 4 2 7 1	0			11	15
4	12	14	9 8 3 6 7 5 4 1 0 2	14			9	13
5	9	13	9 3 8 6 5 7 4 0 1 2	1			13	15
6	1	13	3 6 9 5 9 8 4 2 7 1	7			8	14
7	9	13	9 3 8 6 5 7 4 0 1 2	5		80% the parents are chosen from 'elite ' by using crossover and mutation operation	14	6
8	8	5	7 8 9 1 4 5 2 6 3 0	2			12	15
9	11	15	9 8 3 6 5 7 4 0 1 2	12			13	15
10	5	7	9 8 3 6 5 7 4 0 1 2	15			13	15
11	4	6	9 3 8 6 5 7 4 0 2 1	6			5	13
12	6	14	3 9 6 5 8 0 4 7 2 1	3			13	11
13	13	15	9 8 3 6 7 5 4 1 0 2	8			13	15
14	9	15	9 3 6 8 5 4 7 0 2 1	10			12	15
15	8	11	9 8 3 6 5 7 4 0 1 2	11			15	7

Generation no. 2

Order of population	The chromosome of population: Weight Long Processing Time (wlpt)	The chromosome of population: Weight Short Processing Time (wspt)	Task sort in order of fitness	The chromosome of population was ranked in order of fitness	The parent in population was selected by "elite"		Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Long Processing Time (wlpt)	Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Short Processing Time (wspt)
0	13	5	9 8 3 6 7 5 4 1 0 2	13	13	20% are copied to the next generation without modification	13	15
1	11	15	9 8 3 6 5 7 4 0 1 2	10	10		13	15
2	12	14	9 8 3 6 7 5 4 1 0 2	9	9		13	15
3	11	15	9 8 3 6 5 7 4 0 1 2	5		80% the parents are chosen from 'elite' by using crossover and mutation operation	7	15
4	9	13	9 3 8 6 5 7 4 0 1 2	0			6	14
5	13	15	9 8 3 6 7 5 4 1 0 2	14			13	9
6	8	14	9 3 6 8 5 4 7 0 2 1	8			13	15
7	14	6	7 8 1 4 9 2 5 6 9 3	3			15	10
8	12	15	9 8 3 6 5 7 4 0 1 2	2			12	14
9	13	15	9 8 3 6 7 5 4 1 0 2	1			12	14
10	13	15	9 8 3 6 7 5 4 1 0 2	12			13	15
11	5	13	3 9 6 5 8 0 4 7 2 1	15			13	13
12	13	11	8 9 7 4 5 1 6 3 2 0	6			12	15
13	13	15	9 8 3 6 7 5 4 1 0 2	4			13	15
14	12	15	9 8 3 6 5 7 4 0 1 2	7			13	15
15	15	7	7 8 1 4 9 2 5 6 0 3	11			13	13

Generation no. 3

Order of population	The chromosome of population		Task sort in order of fitness
	Weight Long Processing Time(wlpt)	Weight Short Processing Time(wspt)	
0	13	15	9 8 3 6 7 5 4 1 0 2
1	13	15	9 8 3 6 7 5 4 1 0 2
2	13	15	9 8 3 6 7 5 4 1 0 2
3	7	15	3 9 6 5 8 0 4 7 2 1
4	6	14	3 9 6 5 8 0 4 7 2 1
5	13	9	8 7 9 4 1 5 6 2 3 0
6	13	15	9 8 3 6 7 5 4 1 0 2
7	15	10	8 7 9 4 1 5 6 2 3 0
8	12	14	9 8 3 6 7 5 4 1 0 2
9	12	14	9 8 3 6 7 5 4 1 0 2
10	13	15	9 8 3 6 7 5 4 1 0 2
11	13	13	9 8 7 6 5 4 3 1 2 0
12	12	15	9 8 3 6 5 7 4 0 1 2
13	13	15	9 8 3 6 7 5 4 1 0 2
14	13	15	9 8 3 6 7 5 4 1 0 2
15	13	13	9 8 7 6 5 4 3 1 2 0

Generation no. 4

Appendix for figure (10)

Order of population	The chromosome of population: Weight Long Processing Time(wlpt)	The chromosome of population: Weight Short Processing Time(wspt)	Task sort in order of fitness	The chromosome of population was ranked in order of fitness	The parent in population was selected by "elite"	Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Long Processing Time(wlpt)	Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Short Processing Time(wspt)
0	0	7	3 0 1 4 2	4	4 — 20% are copied to the next generation without modification	12	14
1	0	12	3 0 1 4 2	7	7	9	13
2	7	13	3 0 1 4 2	6	6	14	15
3	5	11	3 0 1 4 2	2	80% the parents are chosen from 'elite ' by using crossover and mutation operation	12	15
4	12	14	3 0 1 4 2	3		12	13
5	8	4	2 4 1 0 3	5		9	14
6	14	8	2 4 1 0 3	1		6	11
7	9	13	3 0 1 4 2	9		4	6
8	3	7	3 0 1 4 2	8		1	14
9	11	0	2 4 1 0 3	0		12	13

Generation no. 1

Order of population	The chromosome of population: Weight Long Processing Time (wlpt)	The chromosome of population: Weight Short Processing Time (wspt)	Task sort in order of fitness	The chromosome of population was ranked in order of fitness	The parent in population was selected by "elite"		Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Long Processing Time (wlpt)	Created Next generation by 20%+(80% after execute crossover and mutation operation): Weight Short Processing Time (wspt)
0	12	14	3 0 1 4 2	2	2	20% are copied to the next generation without 80% the parents are chosen from 'elite ' by using crossover and mutation operation	14	15
1	9	13	3 0 1 4 2	3	3		12	15
2	14	15	2 4 1 0 3	0	0		12	15
3	12	15	2 4 1 0 3	9			12	15
4	12	13	3 0 1 4 2	4			14	7
5	9	14	3 0 1 4 2	5				15
6	6	11	3 0 1 4 2	1			12	5
7	4	6	3 0 1 4 2	6			14	9
8	1	14	3 0 1 4 2	8			15	13
9	12	13	3 0 1 4 2	7			14	1

Generation no. 2

Order of population	The chromosome of population: Weight Long Processing Time (wlpt)	The chromosome of population: Weight Short Processing Time (wspt)	Task sort in order of fitness
0	14	15	3 0 1 4 2
1	12	15	3 0 1 4 2
2	12	15	3 0 1 4 2
3	12	15	3 0 1 4 2
4	14	7	2 4 1 0 3
5	12	15	3 0 1 4 2
6	12	5	2 4 1 0 3
7	14	9	2 4 1 0 3
8	15	13	2 4 1 0 3
9	14	1	2 4 1 0 3

Generation no. 3

REFERENCES

Arroyo, D. O. 1997. STS A Simple Tool for Scheduling. School of Computer Science, McGill University, http://www.cs.mcgill.ca/~cs251/oldcourses/1997/

Auyeung, A., Gondra, I. and Dai, H.K. 2003. Multi-Heuristic List Scheduling Genetic Algorithm for Task Scheduling. ACM. http//www.cs.okstate.edu/~wingha/sac03-ga.pdf

Esquivel, S. C., Gatica, C. R. and Gallard, R.H. 2000. A genetic Approach Using Direct Representation of Solutions for the Parallel Task Scheduling Problem. Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

Garey, M. R., Graham, R. L. and Johnson, D. S. 1978. Performance Guarantees for Scheduling Algorithms. *Operations Research Society of America*, Vol. 26, No. 1.

Gonzalez, M. J. 1977. Deterministic Processor Scheduling. *ACM. Computing Surveys*, Vol. 9, No. 3.

Grajcar, M. 1999. Genetic List Scheduling for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System. http://portal.acm.org/ citation.cfm?id=309931&coll.

Hwang, K. and Briggs, F.A. 1984. *Computer Architecture and Parallel Processing*, McGraw-Hill International Editions, Computer science series, New York, 590-604.

http:// en.wikipedia.org /wiki /genetic algorithm .

Konar, A. 2000. Artificial Intelligence and Soft Computing Behavioral and Cognitive Modeling of the Human Brain, CRC press, Inc., Boca Raton London New York Washington, D.C., 450.

Mahmood, A. 2000. A Hybrid Genetic Algorithm for Task Scheduling in Multiprocessor Real-Time Systems. http://www.ici.ro/ici/revista/sic2000-3/art05.html

Mitchell, M. 1998. An Introduction to Genetic Algorithms, MIT press, London.

Muntz, R. R. 1969. Optimal Preemptive Scheduling on two Processor Systems. *IEEE Transactions on computers*, Vol. C-18, No.

Nossal, R. 2000. An Evolutionary Approach to Multiprocessor Scheduling of Dependent Task. http:// www.citesser.ist.psu.edu/context/377390.

Ramamoorthy, C. V., Chandy, K. M. and Gonzalez, M. J. 1972. Optimal Scheduling Strategies in Multiprocessors System. *IEEE Transactions on Computers*, Vol. C-21, 2.

Thiebaut, D. 1995. Problem Decomposition on a multiprocessor Network. Parallel Programming in C. for the Transputer, chap. IX. http// cs.smith.edu/~thiebaut/ transputer/chapter9/chap9-1.html

Yun, D. Y. Y. 1997. Task Assignment for Parallel and Distributed Computing Advanced in Parallel and Distributed Computing. Shanghai, China. http//csd1.computer.org/comp/proceedings/apdc/1997/7876/00/78760270abc.htm

خوارزمية جينية تخمينية لجدولة المهام المستقلة

*إسراء نذير الكلاك، ورفية زيدان شعبان**

ملخص

تناول البحث دراسة مفهوم الخوارزمية الجينية في جدولة المهام المستقلة للمعالجات المتعددة، من خلال خوارزمية جينية تخمينية مقترحة ناتجة عن دمج خوارزميتين تقليديتين لجدولة المهام المستقلة، للحصول على جدولة مثلى وبأقل زمن للتنفيذ (طول الجدولة). أثبت الجانب العملي كفاءة الخوارزمية الجينية المقترحة عن طريق استخدام إحدى الخوارزميتين.

الكلمات الدالة: الخوارزمية الجينية، جدولة المهام.

* فرع العلوم الأساسية، كلية التمريض؛ ووحدة الحاسبة الالكترونية، كلية الطب، جامعة الموصل. تاريخ استلام البحث 2007/11/25، وتاريخ قبوله 2008/7/8.