technische universität dortmund

department of computer science

TECHNICAL REPORTS IN COMPUTER SCIENCE
Technische Universität Dortmund

DISTRIBUTED RIMEP2: A COMPARATIVE STUDY BETWEEN A HIERARCHICAL MODEL AND THE ISLANDS MODEL IN THE CONTEXT OF REVERSIBLE CIRCUITS DESIGN

Fatima Zohra Hadjam
UNIVERSITY OF DJILLALI LIABES,
SIDI BEL ABBES, ALGERIA

Claudio Moraga
LEHRSTUHL INFORMATIK I
LOGIK IN DER INFORMATIK

Number 853
Mai
2016

Technische Universität Dortmund – Fakultät für Informatik
Otto-Hahn-Str. 14, 44227 Dortmund

Distributed RIMEP2: a Comparative Study between a Hierarchical Model and the Islands Model in the context of reversible circuits design.

Fatima Zohra Hadjam[1] Claudio Moraga[2]
[1]University of Djillali Liabes, Sidi Bel Abbes, Algeria.
[2] TU Dortmund, 44221, Dortmund, Germany
fatimazohra.hadjam@univ-sba.dz claudio.moraga@tu-dortmund.de

Abstract

A distributed hierarchical evolutionary system, named DRIMEP2, for the design of reversible circuits was earlier successfully introduced. In the present work we extend the concept of distributed evolutionary design algorithm, enlarging DRIMEP2 to a family of distributed systems including the hierarchical model, the Island Model, and two hybrid architectures: one comprising a hierarchical model with islands at the lower level, and another one consisting of islands of hierarchical models. A set of 17 randomly chosen 4-bit reversible benchmarks has been evolved under similar parameter environments for the four studied systems. For each benchmark, 100 independent runs were realised and statistics such as "number of successful runs", "average quantum cost", "average gate count" and "total execution time" were considered in the comparison. The results show that in most cases the straight hierarchical model and the hierarchical model with islands of workers are the best in terms of "quantum cost" and "successful runs over 100 runs", although all four distributed DRIMEP2 systems obtained a close performance.

1 Introduction

Bennett showed, in [1], that in order to limit power dissipation, it is necessary that all the computations are performed in a reversible way. This started new research field towards what today is called "Reversible Computing". The design of reversible circuits is only one step within the long chain for the construction of a reversible computer particularly considering the expected further technological advances towards quantum computing.

A reversible circuit consists of a cascade of reversible gates (elementary reversible building blocks). Both a reversible circuit and a reversible gate have the following features: the vectors of inputs and outputs are one-to-one mapped; thus, the vector of input states can be always reconstructed from the vector of output states. They realise bijections. They should be cycle free (i.e. no feedback loops are allowed) and fan-out free (i.e. every gate output can be used only once: either as a circuit output or as an input to a single other gate). Some comprehensive reviews about common reversible circuit synthesis approaches are presented in [2] and [3]. Some recent optimisation results may be found in [22].

RIMEP, a linear genetic programming approach to design reversible circuits, was introduced in [4] and [5]. A further improved version under the name of "RIMEP2" was investigated in [6] and [7].

The various constraints imposed by the reversibility, as mentioned above, and the very large problem space make the design of circuits very difficult. RIMEP2 as an evolutionary approach has already obtained quite satisfying results by evolving reversible circuits from scratch without relying on libraries, databases, or a hash-table of partial solutions, working with a library of gates comprising NOT, CNOT and mixed polarity controlled Toffoli and Peres gates. However to enlarge the potential of RIMEP2 to exploit when exploring the search space, we propose in this report to distribute RIMEP2. The idea is far from just enlarging the values of certain parameters such as the population size, chromosome length or number of generations and then distribute RIMEP2 upon multiple processes to speed up the system. Our goal is basically to allow the method to explore more areas of the search space without going through a blind search, and to exploit this same search space without being trapped in a "plateau" (premature convergence). Under the name DRIMEP2 (where the "D" stands for "distributed") we introduce a family of evolutionary systems for the synthesis of reversible

circuits. We first review a hierarchical structure [20], where the processes are separated in "workers" acting as independent and not communicating "explorers" and a "main unit" acting as an "exploiter", with an information flow only from the explorers to the exploiter. Furthermore, based on the "island model" (see e.g. [12], [13]) we implemented a system where the islands run RIMEP2 processes, and another one where the islands are hierarchical structures. Finally we consider a hierarchical structure where the workers are island models. A comparative analysis of performance of these systems will be done based on a set of benchmarks for reversible circuits.

2 The proposed hierarchical model for RIMEP2

Introduction

RIMEP2 is linear genetic programming-based system [7]. It consists of a population of individuals each representing a potential solution for the problem being evolved, which in the present study is a reversible circuit. Each individual is structured as a cascade of reversible multiple input-output gates. Usually, the parameters such as the size of the population, the length of the chromosome and the number of generations are quite complex to fix since they are problem dependent. Several approaches can be found in the literature to tune this kind of parameters. One can cite "trial-and-error" [8], parameter tuning [9] and parameter control [10]. A systematic tuning approach for RIMEP to solve reversible design problems based on the DOE (Design Of Experiments) method was reported in [11].

When evolving a given circuit, besides finding a correct solution (matching the specification of the circuit), RIMEP2 aims to minimize the quantum cost and the gate count of the circuit. (On how to calculate the quantum cost and gate count, please refer to e.g. [19] and [21]).

Increasing the population size may certainly increase the probability to find an optimal solution but on the negative side, it may lead to a blind search. A strong increment of the number of generations to allow a long search will slow down the system.

Limiting the population size and the number of generations may lead to a premature convergence and the algorithm will become trapped in a plateau.

Hierarchical model based on RIMEP2

The RIMEP2 algorithm is reproduced in one "main unit" and multiple "workers units". Herein, we talk about a two levels hierarchy. If the number of levels is higher then, every worker unit may in turn be a "sub-main unit" of another hierarchy. Figure 1 shows the new proposed hierarchical structure with 2 and 3 levels. In Figure 1.b, the nodes at the second level are workers for the upper level and sub-main units for the lower level. The words "hierarchical" and "hierarchy" refer to the structure in which these units are assembled and to the way they are communicating. The proposed architecture differs from the one (often called "master-slave") found widely in the literature (see e.g. [12]), where the master is an evolutionary algorithm which stores the population, executes the genetic operations and distributes the individuals fitness calculation upon multiples slaves which in turn send the results back to the master (see Figure. 2). The novelty of the proposed architecture may be summarized as follows:

- Every unit is an "instance of RIMEP2". The parameter setting may however differ.
- The communication is done in "one way": from the workers units to the main unit. It is not bi-directional.
- The workers evolve isolated from each other (i.e. independently and without information exchange). Their evolution periods, on given areas of the problem search space, are short before moving to others randomly selected areas. This is realized by a total/partial re-initialization of their populations using different randomly generated seeds. In the present paper, we call this event a "jump". The goal of these jumps is to allow the workers to play the role of explorers. One can ask the following question: why do we need to multiply the units and not just do it at the level of one instance of RIMEP2? The answer is simple: making a re-initialization followed by a short evolution makes it possible to improve the quality of the new introduced individuals. Additionally, the fact that all units are working in parallel does not affect the total execution time and by multiplying the workers units it is possible to explore a large amount of regions of the search space in parallel.

- The parameter setting may differ from one population to another such as using only crossover or only mutation or both, using different rates of crossover and mutations or may be using different types of selection. Different libraries can be assigned to different populations with different random generator seeds to assure a maximum of diversity. A library consists of the different gates used as building blocks during the evolution of a reversible circuit. The chromosome encoding is kept fixed in all the units in order to avoid pre/post- processing during the migration event (see below).
- Both the workers and the main unit(s) are evolving their populations. When the workers reach the limit of the pre-fixed number of generations, the migration event will happen, but always in one way: selected individuals will be sent from the workers to the main unit. Furthermore, each worker will undergo a partial/total re-initialization (jump) to possibly explore a new area of the search space. The main unit will receive the migrants from its workers and continue its evolution. In this way the diversity is supported. The whole system will stop when the limit of the maximum number of generations of the main unit is reached. The best individuals of the main unit will be considered as the solution delivered by the system.
- Additionally to RIMEP2's parameters, we mention four important parameters:
    - Who is migrating? The best individuals in term of fitness values (for the fitness computation see [6] and [7]). We have imposed a restriction that a main unit cannot receive similar individuals (at the same time) even from different workers.
    - Who is/are replaced? The evident way is to replace the worst individuals. According to our experiments, an individual is categorized as worse if it has the highest fitness (our aim is to minimize the fitness) but it does not mean that this individual cannot hold good genetic material capable to improve the best solution. So first, the duplicates (individuals who have similar phenotypes) are replaced and then if necessary, the worse individuals are replaced by migrants (if the migrant is better than the worse individual in term of fitness).
    - How many times should the migration occur (called "frequency")? It is determined by the ratio: $\frac{\text{number of generations of the main unit}}{\text{number of generations of the worker}}$ (fixed to 10 in the performed experiments).
    - How many migrants should migrate (called "rate"). In general it should not exceed a percentage (%) of the whole population of the receiver. In our experiments, we decided that the value of this rate should be more or less the same in the four considered models (12 ± 1 migrants).

These parameters are set experimentally. More details about the whole set of parameters are given in the in the following section.

3 The island model based on RIMEP2

The island model is also called distributed EAs, coarse-grained model, or multi-deme model. Islands are regarded as subpopulations that make up together the population of the whole island model. In the present case, each island will run a RIMEP2 process. Islands evolve independently for a period of time (called "epoch") after which, they exchange individuals (solutions) in a process called "migration". If the epoch is constant the migration is said to be "synchronous" ([12], [13] and [16]).

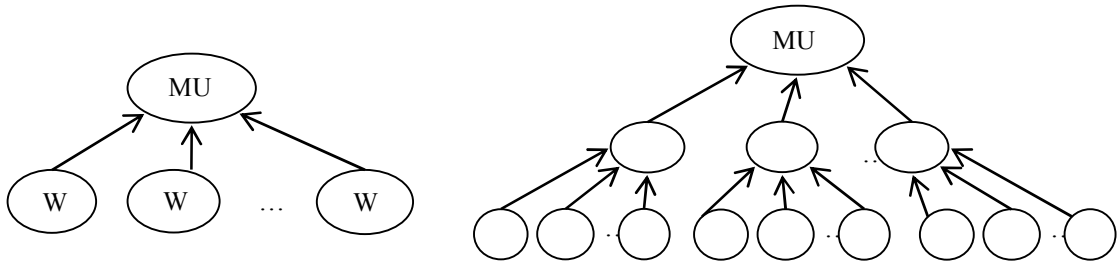


Figure 1: The proposed Hierarchical architecture. (a) A 2-levels hierarchy. (b) A 3-levels hierarchy. "MU" means Main Unit and "W" means Worker. The information flows always from a worker to its main/sub-main unit.

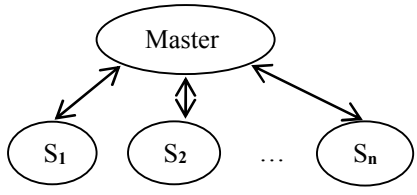


Figure 2: The master/slave Model. "S" indicates Salve. The exchange is done from the master to the slave (as a request to calculate the fitness of a set of individuals) and back from the slave to the master (returning the requested fitness values).

The island model behaviour follows a migration policy, which comprises:

- The migration topology. It is represented by a directed graph. Each node symbolizes an island and each directed edge connects two islands. Among the most famous topologies, one may cite the "ring" and "torus" topologies (see Figure. 3). In the present case, a ring of "RIMEP2 islands" is considered.
- The migration interval. Defined above as epoch. It is the time interval between two successive migrations. Its reciprocal is called migration frequency. Frequent migrations may lead to stagnation due to the increasing number of similar individuals whereas rare migrations yield isolated islands.
- The migration rate. Also known as migration size, determines how many migrants are moving between neighbouring islands. It is proportional to the population size of each island.
- Who is migrating and who is replaced? The candidates to migrate and the individuals to be replaced may be selected based on the fitness (best or worst), randomly or using the same selection operator used within the island. In addition, migrants can be recombined with individuals present in the island before selection.

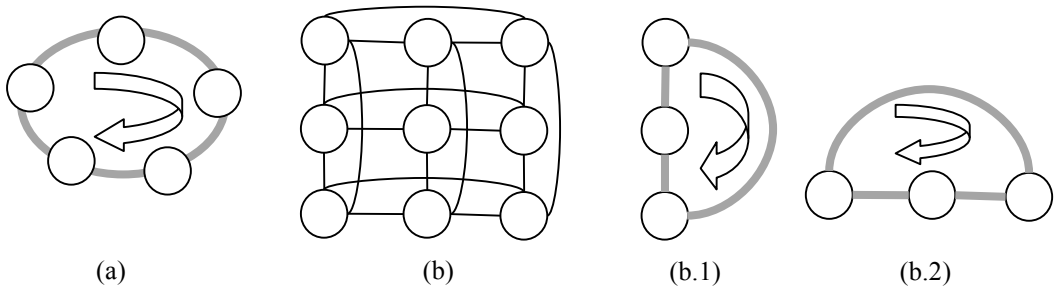


Figure 3: Common Topologies of the island model. (a) Ring topology. (b) Torus or mesh topology. (b.1) and (b.2) indicate the communication direction in the horizontal and the vertical rings of the torus topology.

4 Hybrid 1: A hierarchical model with islands of workers

This model consists of a hierarchy of separated islands. The structure of hybrid 1 is shown in Figure 4. The workers are grouped in a set of mini-island models with "unidirectional ring" topology. The groups are working independently from each other. No exchange is permitted among the groups. Two kinds of migrations may occur:

- Inside each group following the migration policy of the island model described above.
- Form the representative of each group (should be one of the members of the group) to the main unit. If the number of levels of the hierarchy exceeds two, then the groups will be formed of workers of the lower level. Recall that role of the workers is to explore and by this type of hybridization one aims to accelerate the exploitation inside the group taking advantage of the cooperation inside the island model.

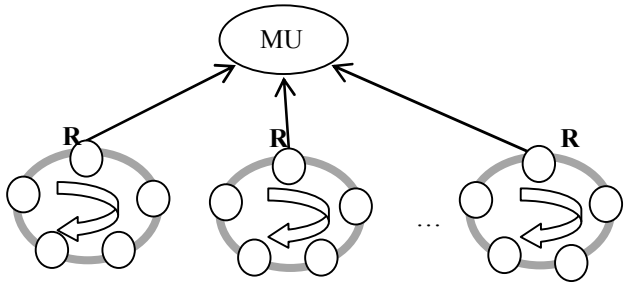


Figure 4: the topology of hybrid 1. "R" stands for "Representative" of each group of islands.

5 Hybrid 2: Ring of hierarchical models

This model consists of multiple parallel hierarchies wherein the main units constitute an island model. See Figure 5. Each hierarchical model is working isolated from the others. The main units exchange individuals in a migration process every period of time. If the number of levels of the hierarchies exceeds two, then island models are formed around the sub-main units also (see Figure 6).

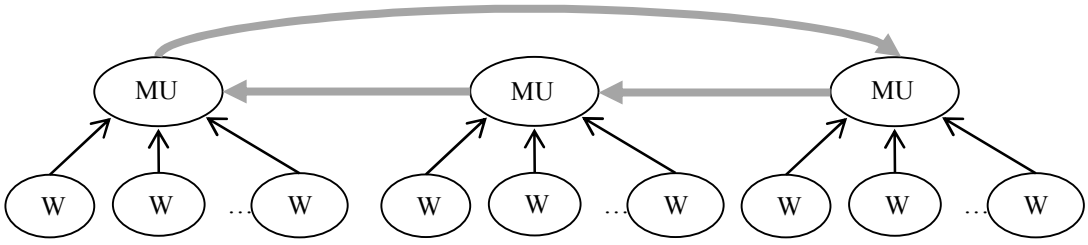


Figure 5: The topology of Hybrid 2. Island model at the upper level and hierarchical model at the lower level.

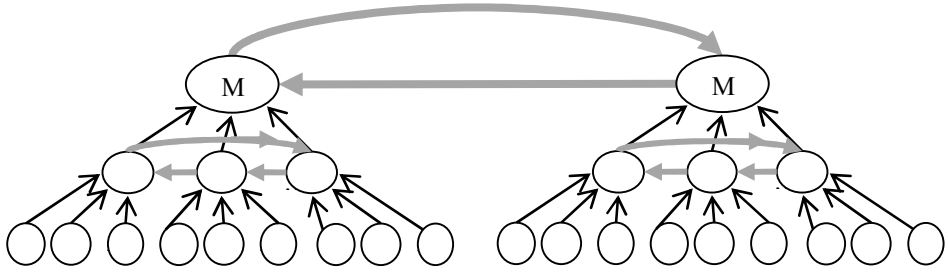


Figure 6: The number of levels exceeding one. Both the main units and the sub-main units constitute island models showed by the rings in gray arrows.

6 Experimentation and interpretation of the result

6.1 Parameter setting

In order to make a fair comparison among four different DRIMEP2 architectures the following main boundary conditions were fixed: a common number of processing units (with an eventual tolerance of 1) and a common number of individuals for every processing unit in order to achieve similar processing times per unit. (The transmission time required for the migration of individuals form one unit to another is, in this context, not relevant.)

Besides the parameters related to the sequential RIMEP2 which were tuned following previous experiments [16] and are given in Table 1, all the processing units are using the same encoding and length for the chromosome. The number of individuals, the total size of the populations, is kept close to 480. Additional parameters for the proposed hierarchical models and island model are considered, keeping an equivalent parameter environment for all of them. For example, the amount and the time interval of transferred migrants from one population to another one were preserved. The number of transferred migrants was fixed to 11 or 12 depending to the used topology (see Table 2; additional information is given later). The migrant candidates are selected according to their fitness (the best) and will replace first the duplicated individuals and then the worst individuals at the receiving unit. This is applied to the four models. All the used hierarchies are built on two levels (i.e. one main unit and n workers). Furthermore all the used island models follow a unidirectional ring. Except for the RIMEP2-island model, the partial re-initialization, called "jump", considered as a key to explore in parallel multiple areas of the search space was fixed to 50% (i.e. 50% of the individuals of a population are replaced by new generated ones using a different seed. (We mean by seed the seed of the digital generator of random numbers). More details are shown in Table 2. Notice that Table 2 is called "DRIMEP2 parameter settings" since it covers all four presented models of Distributed RIMEP2. As mentioned in Table 1, the maximum number of generations is fixed to 10000. In order to keep the migration occurring 10 times in the different discussed models, the maximum number of generations in the case of the workers (hierarchical and hybrids) is fixed to 1000.

Table 1: RIMEP2 parameter setting.

RIMEP2 parameter setting

Parameter	**Value**
Crossover probability	0.7
Mutation probability	0.01
Chromosome length	15
Population size for the sequential version	480
Population size for the version distributed	$40 = \frac{480}{12\ processing\ units}$
Max-number of generations	10000
Selection type	Tournament with size = 2
Crossover type	One-cut crossover

Table 2: DRIMEP2 parameter setting

Distributed RIMEP2 parameter setting

Hierarchical model

Parameter	**Value**	
Number of levels	2	A total of 12 processors
Workers attached to the main unit	11	
Frequency of migration	10 times per run (every 1000 generations)	
Number of migrants	Each worker is sending 1 migrant (a total of 11)	
Percentage of reinitialization	50%	

Island model

Parameter	**Value**
Number of islands	12
Frequency of migration	10 times per run (every 1000 generations)
Number of migrants	Each island is sending 11 migrants

Hybrid 1

Parameter	**Value**	
Number of levels	2	A total of 13 processors
Number of rings of islands	4	
Number of islands inside a ring	3	
Frequency of migration inside the rings (groups) of islands	10 times per run (every 1000 generations)	
Frequency of migration between the representative of each group and the main unit	10 times per run (every 1000 generations)	
Number of migrants inside the ring	2	A total of 12 migrants
Number of migrants from each ring representative to the main unit	1	

Hybrid 2

Parameter	**Value**	
Number of levels	2	A total of 12 processors
Number of independent hierarchies	3	
Workers attached to each main unit	3	
Frequency of migration inside the ring of main units	10 times per run (every 1000 generations)	
Frequency of migration of each worker to its main unit	10 times per run (every 1000 generations)	
Number of migrants inside the ring of main units	3	A total of 12 migrants
Number of migrants from each worker to its main unit	1	

6.2 Benchmarks

To test and compare the four models, 17 4-bit reversible benchmarks have been randomly selected from [14] and [15]. Their compact reversible specifications are given in Table 3. The 4-bit binary strings of the output truth tables are encoded in decimal, for example "5" indicates "0101". Input strings are in lexicographic order.

Table 3: Benchmark specifications Reported benchmarks from [14] and [15]

Name	Specification
4b15g_1	[1,5,0,8,9,11,2,15,3,12,4,6,10,14,13,7]
4b15g_2	[1,9,0,4,10,8,2,11,3,15,5,12,7,14,13,6]
4b15g_3	[3,1,7,13,11,0,8,15,2,5,10,6,9,14,12,4]
4b15g_4	[3,1,11,7,8,0,9,5,2,6,15,13,14,4,10,12]
4b15g_5	[3,5,11,1,8,0,9,7,2,6,14,13,10,4,12,15]
App2.2	[7,14,9,6,11,0,13,2,5,15,10,12,1,4,3,8]
APP2.3	[10,15,0,7,14,9,6,1,13,12,5,3,11,8,4,2]
APP2.8	[12,15,5,8,3,2,1,10,7,14,13,6,11,0,9,4]
App2.11	[8,9,10,2,4,7,6,5,0,15,13,3,12,14,1,11]
nth_prime4	[0,2,3,5,7,11,13,1,4,6,8,9,10,12,14,15]
msaee	[11,3,9,2,7,13,15,14,8,1,4,10,0,12,6,5]
4_49_hwb4	[15,2,3,12,5,9,1,11,0,10,14,6,4,8,7,13]
4_49	[15,1,12,3,5,6,8,7, 0,10,13,9,2,4,14,11]
oc6	[9,0,2,15,11,6,7,8, 14,3,4,13,5,1,12,10]
oc8	[11,3,9,2,7,13,15,14,8,1,4,10,0,12,6,5]
f1	[0,1,2,3,15,10,11,13,9,12,5,4,14,8,6,7]
f2	[0,1,2,3,15,10,11,14,8,7,4,13,6,9,5,12]

6.3 Experiments and results

All the benchmarks were evolved under the same parameter setting (according to Tables 1 and 2). The experiments ran on a cluster of computers (LiDOng cluster of the TU Dortmund University, Germany, see [17] for more details about the total number of nodes, the CPU clock rate and other features.). Each processing unit is a processor in the cluster. In order to compare the four models, some statistics were collected during the runs. We distinguish:

- "The number of successful runs". 100 independent runs have been performed for each benchmark. A run is said to be successful if the evolved reversible circuit matches, totally, the specification of the target reversible circuit (a perfect fit with 0-errors).
- "The best quantum cost among 100 independent runs". As mentioned before the quantum cost is well explained in [7]. The quantum cost consists of the number of elementary reversible gates constituting the whole reversible circuit. Each elementary gate has a quantum cost of one. See [18] and [19].
- "The average quantum cost". The average is calculated based on successful runs only (form 100 runs, we consider only the cases with 0-errors).
- "The execution time over 100 runs". It is calculated in seconds. The algorithm ends when the maximum number of generations is reached. It has been experimentally set to "10000" (the time needed to DRIMEP2 to find at least one solution for each benchmark. See Table 1). The results are given in Tables 4 to 7.

Table 4: Average quantum cost (QC) within 100 runs

Benchmark	Sequential	Hierarchical	Island	Hybrid2	Hybrid1
4b15g_1	54.19	52.64	53.12	54.27	51.64
4b15g_2	40.62	40.79	38.59	40.38	39.18
4b15g_3	39.25	36.25	36.25	35.10	36.27
4b15g_4	47.04	49.09	47.05	50.66	50.27
4b15g_5	39.50	38.77	37.06	37.70	37.13
app2.2	50.04	50.66	54.80	51.11	49.41
app2.3	28.76	25.69	26.90	25.94	26.41
app2.8	64.00	55.38	55.25	54.93	54.08
app2.11	25.98	26.26	25.56	26.03	25.81
nth_prime_4	37.38	33.37	33.30	33.28	32.70
msaee	54.39	51.89	47.33	48.90	43.69
4_49_hwb4	37.40	32.13	30.08	31.55	31.70
4_49	36.36	33.25	32.70	32.82	34.88
oc6	54.27	55.05	56.10	53.66	54.39
oc8	44.79	50.29	45.95	51.91	50.31
f1	25.29	24.10	24.75	24.27	23.58
f2	38.65	28.00	28.33	32.00	30.00
Average	42.23	40.21	39.60	40.27	39.50
S-deviation	10.74	11.25	11.15	11.10	10.54

Table 5: Best quantum cost (QC) within 100 runs

Benchmark	Sequential	Hierarchical	Island	Hybrid2	Hybrid1
4b15g_1	35	36	37	39	37
4b15g_2	32	30	34	32	30
4b15g_3	33	30	33	30	32
4b15g_4	34	33	33	34	34
4b15g_5	33	30	30	30	31
app2.2	36	36	38	35	36
app2.3	21	21	22	22	22
app2.8	35	34	34	35	35
app2.11	24	23	23	24	24
nth_prime_4	28	28	28	27	27
msaee	34	34	34	35	34
4_49_hwb4	27	28	26	28	25
4_49	26	27	27	26	27
oc6	37	34	37	36	36
oc8	35	34	35	34	34
f1	21	21	21	21	21
f2	30	25	25	29	24
Average	30.65	29.65	30.41	30.41	29.94
S-deviation	5.21	4.97	5.58	5.22	5.36

Table 6: Successful runs over 100 runs.

Benchmark	Sequential	Hierarchical	Island	Hybrid2	Hybrid1
4b15g_1	84	92	85	95	97
4b15g_2	39	39	32	32	34
4b15g_3	8	8	8	10	11
4b15g_4	48	45	55	59	64
4b15g_5	6	30	34	27	32
app2.2	91	100	96	100	98
app2.3	58	88	59	87	82
app2.8	94	90	85	89	93
app2.11	95	76	80	74	80
nth_prime_4	48	38	20	32	27
msaee	31	45	24	29	39
4_49_hwb4	30	15	13	11	20
4_49	25	20	23	17	16
oc6	97	97	88	98	94
oc8	28	66	37	65	51
f1	100	100	100	100	100
f2	23	2	3	2	4
Average	**53.24**	**55.94**	**49.53**	**54.53**	**55.41**
S-deviation	**33.37**	**34.69**	**33.46**	**36.13**	**34.77**

Table 7: The total execution time (in seconds) for 100 independent runs.

Benchmark	Sequential	Hierarchical	Island	Hybrid2	Hybrid1
4b15g_1	43360	689	712	686	711
4b15g_2	44180	687	699	681	707
4b15g_3	43820	774	775	773	780
4b15g_4	42210	610	609	603	593
4b15g_5	44620	777	824	787	833
app2.2	43070	707	716	700	715
app2.3	43550	696	740	711	703
app2.8	43780	771	780	767	769
app2.11	43490	727	775	745	726
nth_prime_4	43090	710	750	718	719
msaee	43730	742	769	753	779
4_49_hwb4	40791	625	612	626	629
4_49	43629	766	791	769	778
oc6	42131	694	699	694	692
oc8	43361	704	732	701	706
f1	42792	685	733	708	703
f2	40006	423	420	422	426
Average	**43035.88**	**693.35**	**713.88**	**696.71**	**704.06**
S-deviation	**11.80**	**0.84**	**0.95**	**0.87**	**0.92**

The parameters given in Table 1 allowed the sequential RIMEP2 to find quite optimal solutions (even the best in the case of the benchmark "4b15g_1". See Table 4) and avoiding quick premature convergence due to the high population size fixed to the value of 480. But the negative side was the expensive execution time (a factor of 60) compared to the rest of the models. The hierarchical and the island models, given respectively by Figures 1 and 3, were then experimented. On one hand and according to the penultimate row of Table 4 (values in bold) and in terms of overall quality of the solutions (average quantum cost over 100 of independent runs), the model of islands was the winner (see Figure 7.a. Recall that the average quantum cost is to be minimized), even if the difference is small due probably to the fact that parameter setting was strong enough to permit DRIMEP2 (under different models) to find optimal solutions. According to the results shown in Table 4, among 17 benchmarks the island model outperformed 13 times the sequential model and 10 times the hierarchical model. On the other hand, the best quantum cost solutions and the best number of successful runs were realized by the proposed hierarchical model with less execution time compared with the island model (see the penultimate rows of the Tables 5, 6 and 7). In term of "best quantum cost", the hierarchical model outperformed or matched the island model 16 times among 17 and 12 in term of "successful runs". Refer to Figures 7.b and 8. Recall that the aim is to minimize the former and maximize the later.

To benefit from the strengths of both hierarchical and island models, we combined them in two hybrids "hybrid 1" and "hybrid 2" described in Figures 5 and 6 respectively:

- Hybrid 1. In order to accelerate the process of exploitation for the explorers (workers), we grouped the workers attached to the same main unit, in rings of islands (4 rings of 3 islands each). The results were significant. We noticed a little improvement in terms of average quantum cost (see the penultimate row of Table 4 in the column corresponding to hybrid1). Figure 9.a shows that the shape corresponding to hybrid 1 is slightly inside the shape representing the hierarchical model. At the same time equivalent values were found for the best quantum cost and the number of successful runs in comparison with the hierarchical model, but the negative side was a slight increase in the execution time due probably to the communication time nevertheless it still lower or equal to the execution time of the island model (see Figure 10).

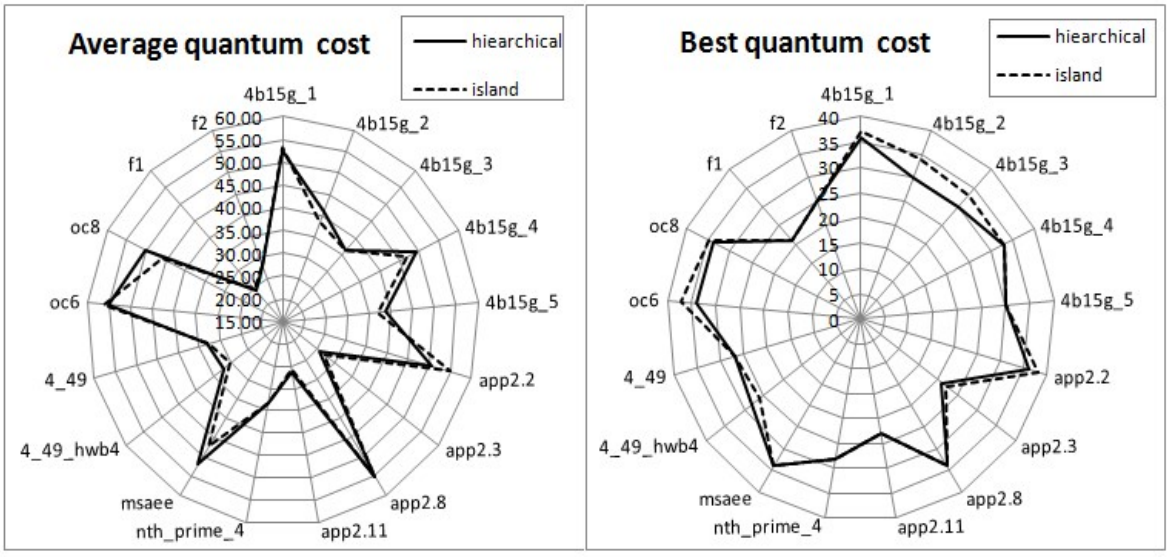


Figure 7: Radar graph representing the comparison between the hierarchical and the island models for the 17 benchmarks in terms of "*average quantum cost*" (a) and "*best quantum cost*" (b).

- Hybrid 2. Although the execution time did not increase, we did not notice a significant improvement in the performance (see Figure 9.b). We concluded that it is probably due to the reduced number (equal to 3) of workers attached to each main unit. Recall that this topology has been set according to the restriction that all the models should have the same number of processing units (with a tolerance of one).

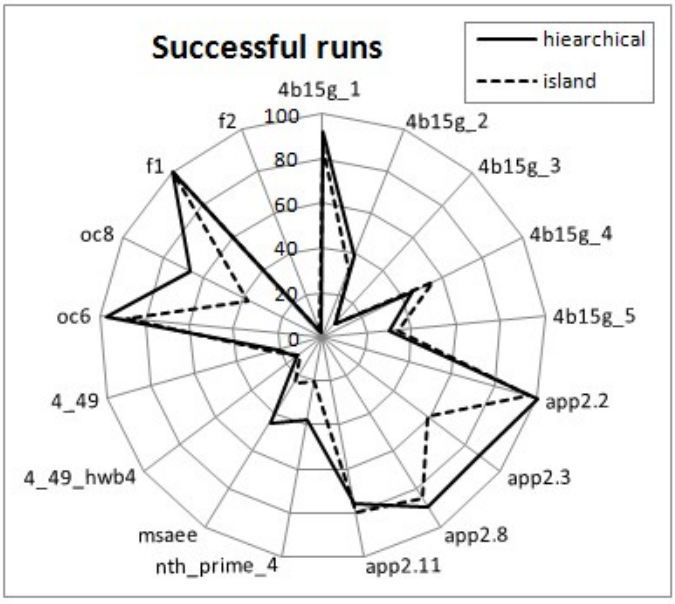
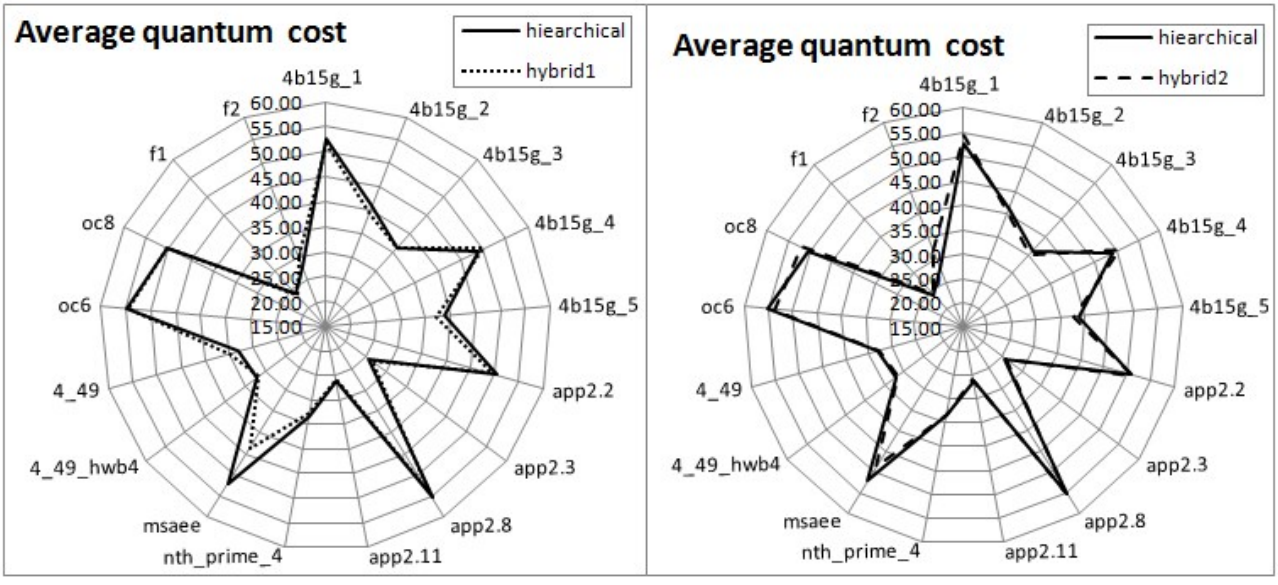


Figure 8: Radar graph representing the comparison between the hierarchical and the island models for the 17 benchmarks in term of "*Number of successful runs*".



(a) **(b)**

Figure 9: Radar graph representing the comparison between the hierarchical and hybrid1 models for the 17 benchmarks in terms of "Average cost". The shape represented by dotted line in (a) is slightly included in the full line represented shape which means that the average cost was partially improved.

(a) **(b)**

Figure 10: the execution time in the case of Hybrid 1 is slightly higher than the execution time of the hierarchical model (a) but still lower or equal than the execution time of the island model (b).

We have, additionally, compared the best quantum cost of the 17 benchmarks reversible circuits evolved by DRIMEP2 using the four proposed models with the best quantum cost corresponding to published results (as of December 2014) found in [14] and [15]. The maximum improvement was equal to 51.16% with an average of 9.22% (see Table 8). It is fair to mention that in the case of the benchmark nth_prime4_inc DRIMEP2 could not match or improve the result of [15], possibly due to the selected limited number of processing units.

Table 8: DRIMEP2 vs [14] and [15]

Benchmark name	**[14] and [15]**	**DRIMEP2**	**Improvement (%)**
	Best QC	**Best QC**	
4b15g_1	39	**35**	**10.26**
4b15g_2	31	**30**	**3.23**
4b15g_3	33	**30**	**9.09**
4b15g_4	35	**33**	**5.71**
4b15g_5	31	**30**	**3.23**
App2.2	35	35	0.00
APP2.3	43	**21**	**51.16**
APP2.8	53	**34**	**35.85**
App2.11	26	**23**	**11.54**
nth_prime4_inc	26	27	-3.85
msaee	34	34	0.00
4_49_hwb4	26	**25**	**3.85**
4_49	28	**26**	**7.14**
oc6	37	**34**	**8.11**
oc8	35	**34**	**2.86**
f1	22	**21**	**4.55**
f2	25	**24**	**4.00**
Average			**9.22 %**

7 Conclusion

In this report we have investigated four different distributed models of RIMEP2 applied to a set of 17 randomly chosen 4-reversible circuit benchmarks [14] and [15]. The results show that in most cases the hierarchical model is the best in terms of “best quantum cost” and “execution time over 100

runs". Possibly due to the constraints of homogeneous resources, all four distributed models have close related average performances, as illustrated by the high overlapping in the radar graphs. The problem dependent performance of evolutionary systems is reflected in the irregular shape of the radar graphs (Figures. 7, 6, and 9), but, at the same time the graphs are very close to each other indicating that the different models reacted in a similar way to the demands of the corresponding benchmarks. The results also show that the sequential RIMEP2, with a large population, was able to evolve optimal results, but at the cost of a high execution time.

8 References

[1] Benett, CH. 1973. Logical reversibility of computation. *IBM J. Res. Dev*. 17, 6, pp. 525-532.

[2] Drechsler, R. and Wille, R. 2011. From truth tables to programming languages: progress in the design of reversible circuits. In *Proceedings of the 41st IEEE International Symposium on Multiple-Valued Logic, ISMVL '11*, pp. 78-85, IEEE Computer Society Press.

[3] Saeedi, S. and Markov, I. L. 2013. Synthesis and optimization of reversible circuits - a survey. *ACM Computing Surveys*, available at arXiv 1110.2574v1.

[4] Hadjam, F. Z., Moraga, C. 2010. RIMEP for Designing Reversible Adders and Multipliers. In *Proc. 2nd workshop on Reversible Computation* (RC), pp. 35-38. Bremen, Germany. July, 2010.

[5] Hadjam, F. Z., Moraga, C. 2010. Evolutionary Design of Reversible Digital Circuits using IMEP: the case of the Even Parity Problem. In *Proc. IEEE World Congress on Computational Intelligence (WCCI)*. pp. 1-6. Barcelona, Spain. July 18-23, 2010. IEEE Computer Society Press.

[6] Hadjam,F.Z., Moraga. C. 2014. RIMEP2, Evolutionary Design of Reversible Digital Circuits. *ACM Journal on Emerging Technologies in Computing Systems*. Special Issue on Computational Synthetic Biology and Regular Papers. Volume 11 Issue 3, Article No. 27, December 2014, ACM New York, NY, USA.

[7] Hadjam, F. Z., Moraga, C. 2014. Introduction to RIMEP2: A Multi-Expression Programming System for the Design of Reversible Digital Circuits. Technical Report ECSC-2014-FSC-02, ISSN 2254 - 2736. European Centre for Soft Computing. Available at arXiv:1405.2226v1.

[8] Wade, H. 2005. Trial and error : An organized procedure. *InTech*, 52(5), pp. 38–42.

[9] Smit, S. K.,. Eiben, A. E. 2010. Parameter tuning of evolutionary algorithms: generalist vs. specialist. In *Proc. 2010 international conference on Applications of Evolutionary Computation, EvoApplicatons'10*, pp. 542–551, Berlin, Heidelberg. Springer-Verlag.

[10] Eiben, A. E. Hinterding, R. and Michalewicz, Z. 1999. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comp*, 3(2), pp. 124–141.

[11] Hadjam, F. Z. 2015. Tuning of Parameters of a Soft Computing System for the Synthesis of Reversible Circuits. *Jr. of Multiple-Valued Logic & Soft Computing*. 24(1-4), pp. 341-386.

[12] Cantu, P., E. 1997. A survey of parallel genetic algorithms. Technical Report, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbana, IL.

[13] Alba, E. and Troya, José M. 1999. A survey of parallel distributed genetic algorithms. *Jr. of Complexity*. 4(4), pp. 31-52.

[14] Younes, A. 2012. Detection and Elimination of Non-trivial Reversible Identities. *International Journal of Computer Science, Engineering and Applications* (IJCSEA), 2, (4), August 2012.

[15] Szyprowski, M. and Kerntopf, P. 2012. A study of optimal 4-bit reversible circuit synthesis from mixed-polarity Toffoli gates. In *Proceedings of the 12th IEEE Conference on Nanotechnology* (IEEE-NANO), pp. 1-6, Birmingham.

[16] Sudholt, D. 2014. Parallel Evolutionary Algorithms. In Janusz Kacprzyk and Witold Pedrycz (Eds.): *Handbook of Computational Intelligence*, Springer.

[17] Lidong. 2009. Dortmund University. http://lidong.hrz.tu-dortmund.de/ldw/index.php/Main Page

[18] Maslov, D. 2012. Reversible logic synthesis benchmarks page. Available at http: //www.cs.uvic.ca/~dmaslov. Last accessed July 2012.

[19] Barenco, A., Bennett, C.H., Cleve, Di Vincenzo, R., D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H. 1995. Elementary gates for quantum computation. *Phys- Rev*. A 52, pages 3457-3467

[20] Hadjam F.Z., Moraga C. 2014. Synthesis of 4-bit Reversible Circuits using Hierarchical Distributed Reversible Multi-Expression Programming. Research Report FSC-2014-01, European Centre for Soft Computing.

[21] Maslov, D. and Miller, D. M. 2007. Comparison of the cost metrics for reversible and quantum logic synthesis. *IET Computers and Digital Techniques*, 1(2):98-104.

[22] Dueck, G.W. 2014. Challenges and advances in Toffoli networks optimization. *IET Computers and Digital Techniques*, 8(4):172-177.

Forschungsberichte
der Fakultät für Informatik
der Technischen Universität Dortmund

ISSN 0933-6192

Anforderungen an: Dekanat Informatik | TU Dortmund
D-44221 Dortmund