Indonesian Journal of Electrical Engineering and Computer Science
Vol. 9, No. 3, March 2018, pp. 789~798
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v9.i3.pp789-798

Bi-objective Scheduling with Cooperating Heuristics for Embedded Real-time Systems

Bendib Sonia Sabrina, Kalla Hamoudi, and Kalla Salim
Computer Science Department, University of Batna 253, Route de Constantine. Fesdis, Batna 05078, Algeria

Article Info

Article history:

Received Jan 5, 2018
Revised Feb 14, 2018
Accepted Feb 28, 2018

Keywords:

Embedded real-time systems
Cooperating heuristics
Bi-objective scheduling
Reliability
Pareto

ABSTRACT

This paper proposes Makespan and Reliability based approach, a static sheduling strategy for distributed real time embedded systems that aims to optimize the Makespan and the reliability of an application. This scheduling problem is NP-hard and we rely on a heuristic algorithm to obtain efficiently approximate solutions. Two contributions have to be outlined: First, a hierarchical cooperation between heuristics ensuring to treat alternatively the objectives and second, an Adapatation Module allowing to improve solution exploration by extending the search space. It results a set of compromising solutions offering the designer the possibility to make choices in line with his (her) needs. The method was tested and experimental results are provided.

Corresponding Author:

Kalla Hamoudi,
Computer Science Department,
University of Batna 253,
Route de Constantine. Fesdis, Batna 05078, Algeria.
Email: hamoudi.kalla@univ-batna2.dz

1. INTRODUCTION

One principle key in distributed real time embedded system design is scheduling. Indeed, strategies that take into account both algorithmic and architectural characteristics to realize an efficient mapping of tasks to processors are crucial. Furthermore, in distributed real time embedded systems, several often conflicting criteria must be taken into account. Bi-objective scheduling deals with real-world problems that involve two conflicting objectives [1, 2, 3, 4]. It is one of the most active research field in the context of distributed real-time systems. Particularly, the Makespan and the reliability have been dealt by several researches.

Makespan and reliability are simultaneously optimized while scheduling a set of independent tasks on a set of heterogeneous processors [5]. The aim is to converge towards Pareto optimal set. In [6], authors propose a spatial allocation based algorithm that works according to two steps, each one optimizing one of the objectives. In [7], a hierarchical approach favoring reliability is adopted by first selecting processors maximizing the reliability and then choosing processor that minimizes the earliest start time. The work presented in [8] extends the list scheduling heuristic [9].

The pair (failure rate, execution time) is used as a basis for choosing the best pair (processor, task). In [10], Makespan and reliability are normalized and combined in a weighted cost function. Processor selection is based on cost function minimization.

The authors of [4] propose a new heuristic for heterogeneous systems that simultaneously optimizes the Makespan and the reliability of an application. A weighted function guides solution choice. In addition, solutions are classified to allow the user adapting the function and then selecting the appropriate solution. The work in [11] presents a bi-objective scheduling heuristic where Makespan and reliability normalization is realized by a compromise function. This one selects for each operation the subset of processors such the

replication of this operation onto those processors maximizes the reliability and minimizes the run-time. A function parameter may be varied to favor one criterion.

In [12], a framework for the bi-objective static multiprocessor scheduling problem considering the Makespan and the reliability is proposed. The global reliability of the system is considered. The Pareto curve is produced for a given instance allowing the user to choose giving advantage to either Makespan or reliability.

In [13], is presented a bi-objective genetic algorithm in which a weighted sum approach is used to deal with the two objectives Reliability and Makespan. Work in [14] proposes and compares two algorithms MOGA and MOEP based on non-dominated ranking, for solving the bicriteria scheduling problem. In addition, a comparision is made with the weighted-sum based bi-objective approach.

Our work concerns bi-objective scheduling in distributed real time embedded systems while makespan and reliability are the considered objectives. Two heuristics cooperate in a hierarchical way to generate solutions, while an adaptation module is used to achieve a best space exploration.

The paper organization is as follows: In section 2, some assuptions concerning system models and objectives are presented. Section 3, recalls the multi-objective otimization and presents bicriteria scheduling as a bi-objective problem. In section 4, the proposed approach is described. Before concluding, section 5 depicts some experimental results assessing our approach.

2. INPUT ASSUMPTIONS

In our work, some assumptions are considered (Section 2.1 and Section 2.2).

2.1. System Description

Distributed real-time embedded systems are composed of two principal parts, which are the software part (the application algorithm) and the hardware part (the distributed architecture). The specification of these systems involve describing the algorithm components (algorithm model), the architecture components (architecture model), and the execution characteristics of the algorithm onto the architecture (execution model).

The application is modeled by a data flow graph, called algorithm graph and noted Alg. Each vertex is a task and each edge is a data dependency.

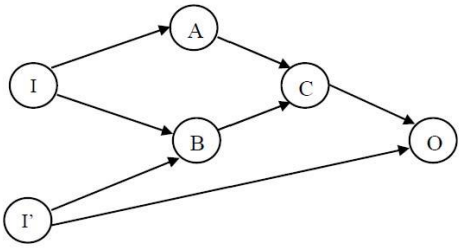

Figure 1A. Algorithm Graph Alg

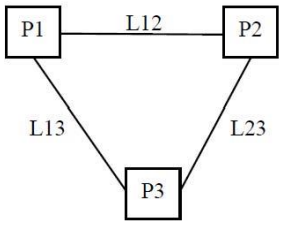

Figure 1B. Architecture Graph Arc

A task of Alg can be either an external input/output task or a computation task. Tasks with no predecessor (resp. no successor) are the input interfaces (resp. output), handling the events produced by the sensors (resp. actuators). The inputs of a computation task must precede its outputs.

Figure 1A is an example of an algorithm graph, with six tasks: I and I' (resp. O) are inputs (resp. outputs) tasks, while A, B, and C are computation tasks. The data-dependencies between tasks are depicted by arrows. For instance the data-dependence A▷C can correspond to the sending of some arithmetic result computed by A and needed by C.

The architecture is modeled by a graph, called architecture graph and noted Arc. Each vertex is a processor, and each edge is a communication link. Figure 1B gives an example of an architecture graph, with three processors P1, P2 and P3, and three communication links.

An execution time Exe is defined for each pair (ti, pj); it represents the worst case execution time of the task ti on the processor pj, expressed in time units. Assuming that processors are heterogeneous, one task could have different execution times on different processors. When a given task cannot be executed on a given processor, the association is expressed by the value "∞". On the other hand, to each pair (di, lj), is

associated a time that expresses the worst case transmission time of the data dependency di on the communication link lj. Furthermore, the intra-processor communication time is supposed to be 0 time unit. For instance, Exe for Alg and Arc of Figures 1A and 1B is given in Tables 1A and 1B.

Table 1A. Execution Times

	I	I'	A	B	C	O
P1	∞	2	6	4	2	4
P2	4	∞	12	8	4	8
P3	3	2	9	6	3	∞

Table 1B. Transmission Times

	I ▷A	I ▷B	I'▷B	I'▷O	A▷C	B▷C
L12	3	2	4	3	1	5
L13	6	5	8	6	2	10
L23	3	2	4	3	1	5

Application and architecture modeling using graphs, is useful for objective definitions (subsection 2.2.).

2.2. Makespan and Reliability Objectives

The makespan or schedule length is the end execution time of the task that is completed last among all tasks. It is defined as follows:

$$M = \max_{p_j} \left\{ \max_{t_i \in p_j} end\,(t_i, p_j) \right\} \tag{1}$$

where, end (ti, pj) is the time at which task ti terminates its execution on processor pj.

A function called *schedule pressure*, calculated from the graph algorithm, is proposed in [15], it is noted $\sigma^{(n)}$ and is defined for each task ti $\in T^{(n)}_{comp}$ (n refering to the heuristic step and comp to the set of competitor tasks i.e those not yet scheduled and whose predecessors are already scheduled) and each processor pj as follows:

$$\sigma^{(n)}\,(t_i, p_j) = S^{(n)}_{best(t_i,p_j)} + \bar{S}^{(n)}_{t_i} \tag{2}$$

where:

$S^{(n)}_{best(ti,pj)}$: the earliest time at which task ti can start execution on processor pj;
$S^{-(n)}_{ti}$: the latest start time from end of ti [15];

Furthermore, execution and transmission times have to considered, hence the following expression:

$$\sigma^{(n)}\,(t_i, p_j) = S^{(n)}_{best(t_i,p_j)} + t_{exe} + t_{comm} + \bar{S}^{(n)}_{t_i} \tag{3}$$

Schedule pression is used to select the best task which mimizises the length of the critical path. That means that schedule pression mimization implies schedule length one. Otherwise, both processors and communication links are subject to failures. According to the model proposed in [12] and considering the occurrence of failures following a Poisson law with a constant parameter λ, the reliability of a processor P (respectively, a communication link L) during the duration d is:

$$Rel = e^{-\lambda d} \tag{4}$$

The reliability of the task or data dependency X placed onto the hardware component C, with an execution time exe(X, C), is then defined as follows:

$$Rel\,(X, C) = e^{-\lambda_c exe(X,C)} \tag{5}$$

Because the reliability depends intrinsically on the duration of the tasks and communications, some technical difficulties raise when using both reliability and makespan as objectives [12].

So, rather than using the usual model of the reliability [16], we use the concept of GSFR (Global System Failure Rate) defined in [12], and noted Λ. The GSFR of a static schedule S, is computed as follows:

$$\Lambda(S)=\frac{-\log R(S)}{U(S)} \text{ with } R(S)=\prod_{(o_i,p_j)\in S} R(o_i,p_j) \text{ and } U(S)=\sum_{(o_i,p_j)\in S} \mathcal{E}xe(o_i,p_j) \quad (6)$$

where U(S) is the total utilization of the hardware resources and R(S) its reliability. The GSFR is the failure rate per time unit of the obtained multiprocessor schedule, seen as if it were a single task scheduled onto a single processor.

3. SCHEDULING AS A MULTI-OBJECTIVE PROBLEM

A multi-objective problem is a problem whose resolution implies multiple objectives consideration. The general optimization problem is defined as follows [17]:

$$Minimize\ f(x) = [f_1(x),\ f_2(x), \cdots f_m(x)]$$

$$\begin{aligned} \text{subject to}\ \ & g_i(x) \le 0 \quad i = 1, 2, \cdots k \\ \text{and}\ \ & h_j(x) = 0 \quad j = 1, 2, \cdots l \end{aligned}$$

where m is the number of objective functions, k is the number of inequality constraints and l is the number of equality constraints. $x \in E^n$ is a design vector, also called decision vector, where n is the number of its independent variables xi. $f(x) \in E^n$ is a vector of objective functions where:\

$$f_i(x) : E^n \longrightarrow E^1$$

$f_i(x)$ are also called objective or criteria. Each point in the design space maps to a point in the objective space but the reverse may not to be true.

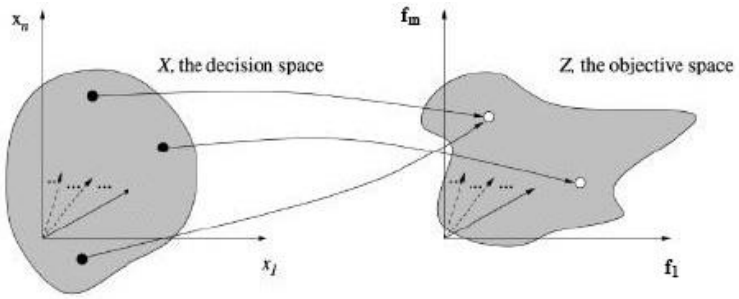

Figure 2. Decision and Objective Spaces

Evaluation of solutions is done by using the Pareto 'dominance' (Pareto 1906). A potentially interesting solution is a solution such as improving one objective can't be done without degrading at least another one. Such solutions constitute the Pareto optimal set. Each solution can be represented by its objective vector in a multi-dimensional space (Figure 2).

Let z and z' be two points of the objective space. Formally, the Pareto dominance on objective vectors is defined as:

A point z is Pareto dominated by a point z':

$$\forall i \in \{1 \cdots m\} \quad z_i^{'} \le z_i \ and \ \exists\, i \in \{1 \cdots m\} \ so \ that \ z_i^{'} < z_i$$

While the concept of dominance is related to objective space, the optimality concerns the decision space.

Pareto optimal solutions constitute what is called the Pareto optimal set while the corresponding objective vectors i.e those ones not dominated, are said to be on the Pareto front. The Neighborhood is another important concept in combinatorial optimization. It is defined as follows:

"The neighborhood of a solution s ∈ S is a subset of configurations or solutions of S that are directly reachable by a given transformation of s. It is noted V (s) and a solution s ∈ V(s) is said to be a neighbor of s."

Particularly, from a given solution, various Neighborhood structures can be established according to various transformations while a transformation is defined as an application:

$$V : S \longrightarrow P(S)$$

where S is a set of solutions and P(S) is a subset of S. Such a concept is useful in search space structuring and in defining the set of solutions that can be reached from a given solution through a series of transformations.

In our work, scheduling is bi-objective and is therefore considered as a bi-objective optimization problem. Solutions in decision space are shedules each of them expressing both task assignment to processors and a given execution order. Each schedule (solution) can be evaluated in the objective space (Figure 3) defined by the considered objectives (subsection 2.2.). The first objective is Makespan (equation (1)) that is defined using the cost function named schedule pressure (equation (3)) while the second objective is GSFR (equation (6)).

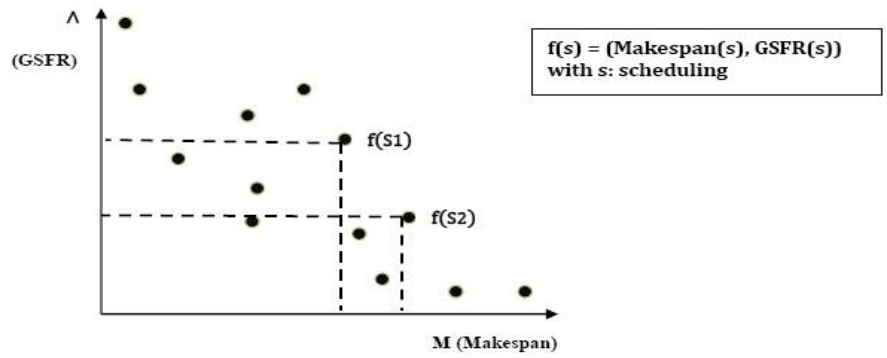

Figure 3. Objective Spaces

Rather than a single solution, a set of solutions (schedules) {s1, s2, · · · } is generated. Each schedule si makes a compromise between the Makespan and the reliability and is expressed by f(si).

4. THE PROPOSED BI-OBJECTIVE APPROACH

Given an algorithm and a target architecture, we aim to produce schedules realizing compromises between two objectives by minimizing Makespan and maximizing reliability.

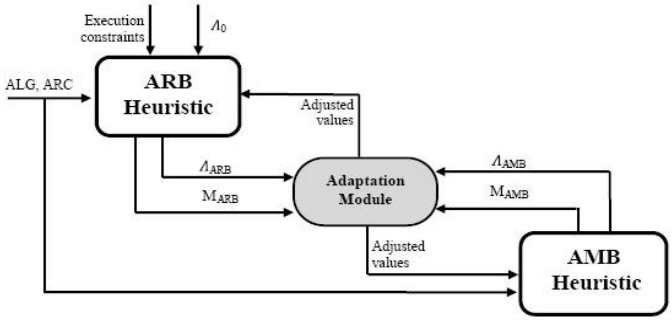

Figure 4. The Proposed Approach

4.1. Approach Principles

Our approach is based on two scheduling heuristics with an adaptation module to improve solution exploration by extending the search space. The two heuristics cooperate while dealing alternatively with the two objectives: Makespan and reliability. We use hierarchical scheduling to minimize Makespan and to maximize reliability by optimizing one objective at a time, i.e, we transform one of objectives into a constraint, which allows the solving of the problem by optimizing the second objective under the constraint of the first one.

As shown in Figure 4, the first heuristic ARB (Adaptive Reliability-based Bi-objective heuristic) has as inputs the algorithm Alg, the architecture Arc and a GSFR (reliability objective) value Λ_0 as a constraint. ARB heuristic execution leads to a solution with two values : M_{ARB} and Λ_{ARB}. Like ARB, AMB (Adaptive Makespan-based Bi-objective heuristic) has as inputs the algorithm Alg, the architecture Arc and the solution produced by ARB heuristic. The Makespan value MARB, produced by ARB, is considered as a constraint in order to maximize reliability. The result is a schedule with two values: M_{AMB} and Λ_{AMB}. Cooperation between our hierarchical heuristics leads to a set of solutions among which the optimal ones according to Pareto optimality are selected.

In the case of too closest compromise values, we propose to introduce an adaptation module to search for neighbors. Note that the fuction f associating decision and objective spaces is not surjective. That means there can be a compromise value in objective space which is not associated with any solution (scheduling) in decision space. For this reason, the adaptation must operate on decision space by creating neighbors of a given schedule, this one (neighbor) having necessarely a compromise value in objective space.

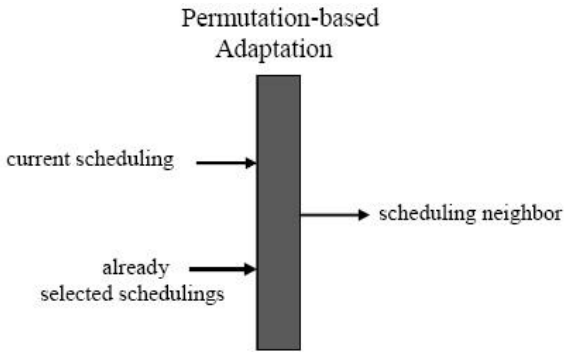

Figure 5. Adaptation Module

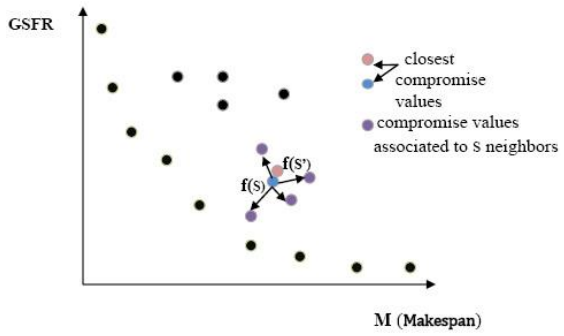

Figure 6. Solution Translation

To adjust a current solution, the adaptation module (Fogure 5) is based on the neighborhood structure concept (subsection 3.). In this work, a permutation-based neighborhood structure is used; it is defined by the following transformation:

$V : S \rightarrow P(S)$ such that: $\forall$ schedule $s \in S$

$V(s) = \{ s' \mid s'$ is a schedule associated to a given permutation of $s\}$

Figure 6 is an example of too closest compromise values f(s) (blue color) and f(s') (pink color) in the objective space. In this case, the adaptation module has the role of extending the search space by applying permutations on s. This one could be replaced by one of its neighbors. By changing the starting constraint for cooperating heuristics and using the adaptation module, a range of solutions is obtained while the ones that suit current needs are selected.

4.2. Scheduling Algorithms

The heuristics ARB and AMB implementing the proposed solution are greedy list scheduling.

ARB Algorithm

Input: $\mathcal{A}lg$, $\mathcal{A}rc$, Λ_0
Output: M_{ARB}, Λ_{ARB}
Begin
Initialise the lists of competitor and scheduled tasks:
$T^{(0)}_{comp} := \{t \in T \mid pred(t) = \emptyset\}$;
$T^{(0)}_{sched} := \emptyset$;
while $T^{(n)}_{comp} \neq \emptyset$ **do**
- Compute the schedule pressure for each task t_i of $T^{(n)}_{comp}$ on each processor p_j such that $\Lambda_i < \Lambda_0$;
- schedule the pair (competitor task t, processor p) such that *schedule pressure* value is minimal;
- Update the lists of competitor and scheduled tasks:
$T^{(n)}_{sched} := T^{(n-1)}_{sched} \cup \{t\}$;
$T^{(n+1)}_{comp} := T^{(n)}_{comp} - \{t\} \cup \{t' \in succ(t) \mid pred(t') \subseteq T^{(n)}_{sched}\}$;

end while
if produced compromise value is too closest to others **then** call Adaptation Procedure; **end if**
end

They cooperate to deal alternatively with the makespan and the reliability objectives. The first heuristic is based on a cost function called schedule pressure (equation (3) in subsection 2.2.) to select the best task which minimizes the length of the critical path. The superscript number, in parentheses in both Algorithms ARB and AMB, refers to the heuristic step.

Constrained by a starting constraint which is a GSFR value, ARB algorithm works as follows:

• **Initialization:** two lists are used,
- $T^{(0)}_{comp}$: is the competitor task list knowing that a task is said to be competitor if it is not yet scheduled and all its predecessors are already scheduled;
- $T(0)_{sched}$: scheduled task list which will constitute, in the end, final schedule.

• **Evaluation: s**tep (1) calculates, for each competitor task its schedule pressure on each processor such the specified constraint (Λ) is satisfied.

• **Selection:** this step (2) is a selection of the best pair (task, processor), thus the one minimizing schedule pressure.

• **Update: s**tep (3) consists in one hand to add the chosen task to scheduled task list and in the other hand to remove the chosen task from competitor task list as well as all its successors such that predecessors of the latters are already scheduled.

Steps 1, 2 and 3 of ARB algorithm are repeated until there are no more competitor tasks. The ARB result is a pair (M_{ARB}, Λ_{ARB}) representing Makespan and GSFR values. During its execution, ARB algorithm may refer to the adaptation module. In a similar manner but constrained by Makespan value produced by ARB algorithm, AMB algorithm aims to optimize this time reliability objective. AMB algorithm works as follows:

AMB Algorithm

Input: $\mathcal{A}lg$, $\mathcal{A}rc$, M_{ARB}
Output: M_{AMB}, Λ_{AMB}
Begin
Initialise the lists of competitor and scheduled tasks:
$T^{(0)}_{comp} := \{t \in T \mid pred(t) = \emptyset\}$;
$T^{(0)}_{sched} := \emptyset$;
while $T^{(n)}_{comp} \neq \emptyset$ **do**
- Compute the GSFR for each task t_i of $T^{(n)}_{comp}$ on each processor p_j such that $M_i < M_{ARB}$;
- schedule the pair (competitor task t, processor p) such that GSFR value is minimal;
- Update the lists of competitor and scheduled tasks:
$T^{(n)}_{sched} := T^{(n-1)}_{sched} \cup \{t\}$;
$T^{(n+1)}_{comp} := T^{(n)}_{comp} - \{t\} \cup \{t' \in succ(t) \mid pred(t') \subseteq T^{(n)}_{sched}\}$;

end while
if produced compromise value is too closest to others **then** call Adaptation Procedure; **end if**
end

Like ARB, AMB may also refer to adaptation module if solution adjustment is necessary. ARB and AMB execution can be repeated as many times as the decision maker decides. In addition, the starting constraint can be modified to create a new process instance. As much for ARB as for AMB, in the case of too closest solutions (in objective space), the adaptation procedure could replace the current solution (in decision space) by one of its neighbors.

In our work, we propose permutation-based adaptation. It consists of generating schedules among which there could be a schedule from which the search will be restarted. Permutation application on s generates schedules that are neighbors of s. However, only schedules satisfying real time constraints are considered. A current schedule s is replaced by one of its neighbors if (i) this one is not yet selected and (ii) its compromise value is not closest to those of already selected schedules and (iii) among s neighbors, it is this which corresponds to the non-dominated compromise value in objective space. In the case of no neighbor satisfies the three conditions, s is saved as current solution.

Adaptation Procedure

Input: current schedule s, already selected schedules
Output: New schedule s_N
Begin
generate the set of schedules by applying permutations on s;
select the subset P_S of schedules satisfying real time constraints;
if ∃ a schedule $s_k \in P_S$ such that
(i) s_k is not yet selected
and
(ii) its compromise value is not closest to those of already selected schedules
and
(iii) among schedules $\in P_S$, $f(s_k)$ is the non dominated one **then**
s_k is considered as the new schedule s_N;
else
s is saved as the new schedule s_N
end if
end

In this way, the search space is extended to the Neighborhood structure produced by permutations. As soos as a solution (schedule) satisfying the three conditions above is found, it is considered as the current schedule. Note that, such a schedule may not be found, in which case current schedule is saved.

5. SIMULATIONS

To evaluate our approach, we have applied the AMB and ARB heuristics to a set of random algorithm graphs and an architecture graph composed of 3, 4, and 5 processors. We use SynDEx to generate the complete set of algorithm graphs. SynDEx is a CAD tool for optimizing and implementing real-time embedded systems (http://www.syndex.org). It has been designed and developed in the INRIA Paris-Rocquencourt Research Center France.

We vary two parameters: the number of task N=20, 40, 60, 80, 100, and the communication-to-computation ratio (CCR), defined as the average communication time divided by the average computation time, CCR=0.1, 1, 10. For each N, 100 graphs have been generated.

The general objective of our simulations is to study the impact of N, P, and CCR on reliability and makespan introduced by AMB and ARB. We compare our heuristics with the heuristic proposed in [11], called RBSA (Reliable Bi-Criteria Scheduling Algorithm) and implemented in SynDEx.

Table 2 shows the makespan and reliability results of executing heuristics on an Alg composed of 50 tasks. ARB-AMB* is the heuristics ARB-AMB without adaptation module.

Table 2. Makespan and Reliability Results for N=50 Tasks

	heuristics		
	RBSA	ARB-AMB*	ARB-AMB
execution 1	644.72, 0.998297	616.51, 0.998539	616.51, 0.998539
execution 2	644.72, 0.998297	608.76, 0.999029	608.76, 0.999029
execution 3	644.72, 0.998297	608.76, 0.999029	**599.70, 0.999312**

As shown in this table, thanks to the adaptation module, ARB-AMB approach gives interesting results and better than RBSA and ARB-AMB*.

Figures 7A, 7B, 8A, 8B, 9A and 9B shows makespan and reliability variations. We can see from the results that our approach performs better than RBSA for all the datasets.

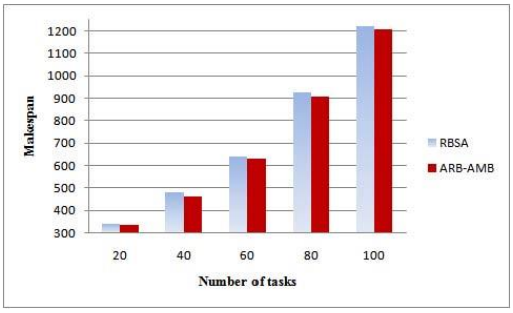

Figure 7A. Impact of N Makespan for CCR=1 and P=4

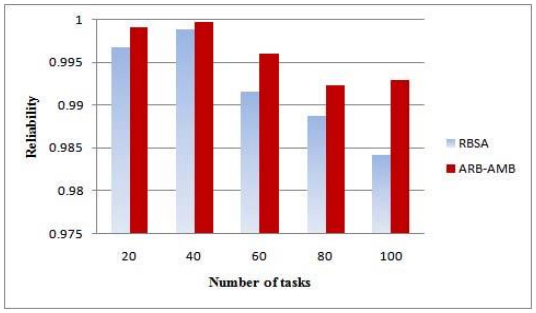

Figure 7B. Impact of N on Reliability for CCR=1 and P=4

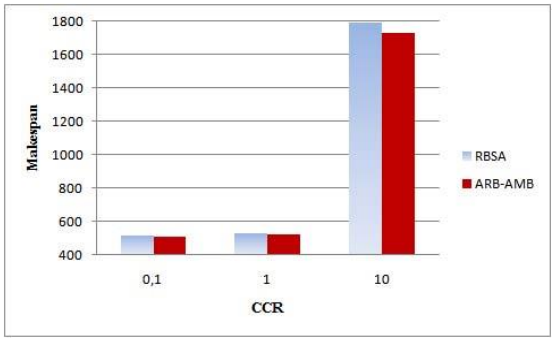

Figure 8A. Impact of CCR on Makespan for N=50 and P=4

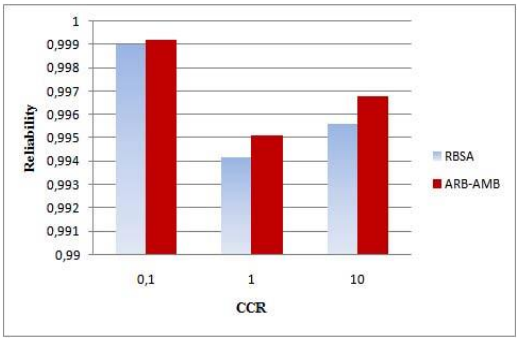

Figure 8B. Impact of CCR on Reliability for N=50 and P=4

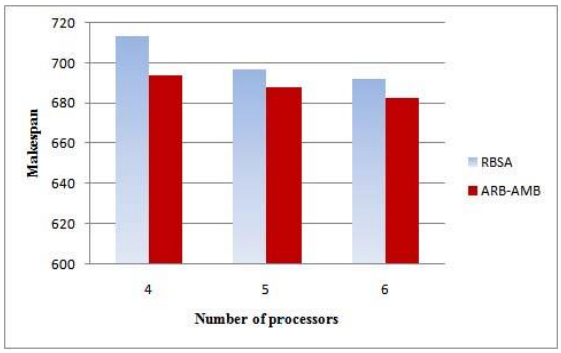

Figure 9A. Impact of P on Makespan for N=50 and CCR=1

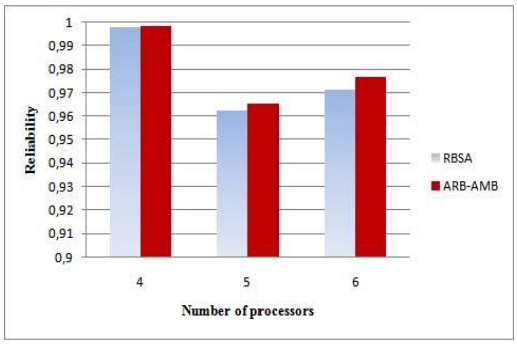

Figure 9B. Impact of P on Reliability for N=50 and CCR=1

6. CONCLUSION

We have proposed a new bi-objective scheduling approach producing automatically a static distributed schedule of a given application Alg on a given distributed architecture Arc. The aim of our approach is to optimize simultaneously two antagonist objectives: system's run-time (Makespan) and reliability. Our approach is based on two cooperating heuristics each of them dealing with an objective. To allow better space exploration, we integrate an adaptation module.

Adaptation is based on the neighborhood concept and is achieved by solution permutation. This allows, in the case of two closest compromise values, to operate on the decision space by generating neighbors. They are schedules among which may be a schedule that is selected in order to improve exploration. As our approach is Pareto-based, a set of compromising solutions (schedules) is produced allowing the designer to select those satisfying the current needs. Simulations results show that the proposed approach performs better than RBSA for all the datasets.

REFERENCES

[1] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Information Sciences*, vol. 379, pp. 241–256, 2017.

[2] Y. Liu, H. Dong, N. Lohse, and S. Petrovic, "A multi-objective genetic algorithm for optimization of energy consumption and shop floor production performance," *International Journal of Production Economics*, vol. 179, pp. 259 – 272, 2016.

[3] S. K. Sonia Sabrina Bendib, Hamoudi Kalla and ChafikArar, "A two-step bicriteria scheduling approach for distributed real time systems," in *International Conference on Electronics, Computer and Computation*, 2013.

[4] I. S. C. Boeres and L. Drummond, "An efficient weighted biobjective scheduling algorithm for heterogeneous systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 37, pp. 349–364, 2010.

[5] E. Jeannot, E. Saule, and D. Trystram, "Bi-objective approximation scheme for makespan and reliability optimization on uniform parallel machines," in *Proceedings of the 14th international Euro-Par conference on Parallel Processing*, ser. Euro-Par '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 877–886.

[6] I. S. A. Girault and D. Trystram, "Reliability versus performance for critical applications," *Parallel and Distributed Computing*, vol. 69, pp. 326–336, 2009.

[7] X. Qin and H. Jiang, "A novel fault-tolerant scheduling algorithm for precedence constrained tasks in realtime heterogeneous systems," *Parallel Computing*, vol. 32, pp. 331–356, 2006.

[8] J. J. Dongarra, E. Jeannot, E. Saule, and Z. Shi, "Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems," in *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, ser. SPAA '07. New York, NY, USA: ACM, 2007, pp. 280–288.

[9] S. H. M. W. H. Topcuouglu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 260–274, 2002.

[10] M. Hakem and F. Butelle, "Reliability and scheduling on systems subject to failures," in *Parallel Processing, 2007. ICPP 2007. International Conference on*, 2007, pp. 38–38.

[11] A. G. I. Assayad and H. Kalla, "A bicriteria scheduling heuristic for distributed embedded systems under reliability and realtime constraints," in *The International Conference on Dependable Systems and Networks, IEEE Computer Society*, 2004.

[12] A. Girault and H. Kalla, "A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 4, pp. 241–254, 2009.

[13] A. Dogan and F. Ozguner, "Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous compuring systems," *The computer journal*, vol. 48(3), pp. 300–314, 2005.

[14] P. P.Chitra, "Comparison of evolutionary computation algorithms for solving bi-objective task scheduling problem on heterogeneous distributed computing systems," *Sadhana, Indian Academy of Sciences*, vol. 36, p. 167U180, 2011.

[15] Y. Sorel, "The algorithm architecture adequation methodology," in *The Massively Parallel Computing systems*, 1994.

[16] J. W. S. Shatz and M. Goto, "Task allocation for maximizing reliability of distributed computer systems," *IEEE Trans. Computers*, vol. 41, pp. 156–168, 1992.

[17] R. T. Marler and J. S. Arora, "Survey of multiobjective optimization methods for engineering, "*Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–3