World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:15, No:5, 2021

Self-Organization-Based Approach for Embedded Real-Time System Design

S. S. Bendib, L. W. Mouss, S. Kalla

***Abstract*—**This paper proposes a self-organization-based approach for real-time systems design. The addressed issue is the mapping of an application onto an architecture of heterogeneous processors while optimizing both makespan and reliability. Since this problem is NP-hard, a heuristic algorithm is used to obtain efficiently approximate solutions. The proposed approach takes into consideration the quality as well as the diversity of solutions. Indeed, an alternate treatment of the two objectives allows to produce solutions of good quality while a self-organization approach based on the neighborhood structure is used to reorganize solutions and consequently to enhance their diversity. Produced solutions make different compromises between the makespan and the reliability giving the user the possibility to select the solution suited to his (her) needs.

***Keywords*—**Embedded real-time systems design, makespan, reliability, self-organization, compromises.

I. Introduction

DESIGN of embedded real-time systems includes a key step which is the mapping of a set of tasks onto a given architecture composed of heterogeneous processors. Furthermore, optimization of conflicting objectives is often needed. These last years, researchers focused on this multi-objective problem taking into account a variety of antagonist objectives.

Schedule length and power consumption are considered in [1]. The proposed algorithm is a two-phase hybrid task scheduling which is based on decomposition of the input task graph by applying spectral partitioning. However, priority is clearly given to power consumption due to assigning each part of the task graph to a low power processor before addressing the schedule length. The work in [2] addresses the problem of reliability and schedule length optimization in an adaptive manner. In fact, this is achieved onto reconfigurable platforms by applying fault tolerance techniques to the running tasks based on the exploration of the Pareto set of solutions in order to select the appropriate ones. Furthermore, a mathematical model of an integer nonlinear multi-objective optimization problem is used for improving the fault tolerance of hardware task graphs, scheduled in partially reconfigurable platforms.

Energy efficiency and timeliness are addressed in real-time embedded systems [3]. The approach is based on Non-dominated sorting genetic algorithm-II while the timing constraints are formulated using type-2 fuzzy sets (T2 FSs). A proactive scheduling algorithm optimizing both performance and power consumption is presented in [4]. Proactivity allows the power management system to assign the devices a low or a high frequency according to the current situation. Embedded systems' design problem is addressed in [5] using a genetic algorithm. Efficient compromises are achieved between reliability, execution time and energy consumption objectives.

An adaptive scheduling algorithm is proposed in [6] for the efficient execution and management of hard and soft real time tasks in embedded IoT systems. The task starvation rate and increasing the CPU utilization are significantly optimized. The work proposed in [7] concerns a self-organization approach for scheduling tasks on heterogeneous processors where Dynamic Crowding Distance is used to reorganize solutions ensuring a better solution diversity.

The paper first presents the given problem as a multi-objective optimization one. In Section III, system models are defined. Section IV depicts the proposed approach while Section V is about experiment results. A conclusion is given in Section VI.

II. Task Mapping: A Multi-Objective Optimization Problem

A multi-objective problem is the one implying two or more conflicting objectives to be optimized. The general optimization problem is defined according to (1):

$$\begin{gathered} Minimize\ f(x) = [f_1(x),\ f_2(x), \dots, f_m(x)] \\ subject\ to\ g_i\ (x) \le 0\ i = 1, 2, \dots, k \\ and\ h_j(x) = 0\ j = 1, 2, \dots, l \end{gathered} \tag{1}$$

where m is the number of objectives functions, k and l are the number of inequality constraints and the number of equality constraints respectively.

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

$$\begin{gathered} \forall\ i\ \in \{1 \dots m\}\ z'_i \le z_i \\ \text{and } \exists\ i \in \{1 \dots m\}\ so\ that\ z'_i < z_i \end{gathered} \tag{2}$$

Based on Pareto dominance concept defined by (2), a potentially interesting solution is the one for which improving one objective cannot be done without degrading at least another one. Each solution can be represented by its objective vector in a multi-dimensional space (Fig. 1).

In the case of the task mapping problem onto processors, each s_i in the decision space expresses a given schedule of tasks on processors while the schedule evaluation is achieved in the objective space through the objective functions f_1 and f_2.

S. S. Bendib is with the Computer Science Department of Banta 2, 05000 Batna Algeria (corresponding author, phone: +213772247158; e-mail: ss.bendib@univ-batna2.dz).

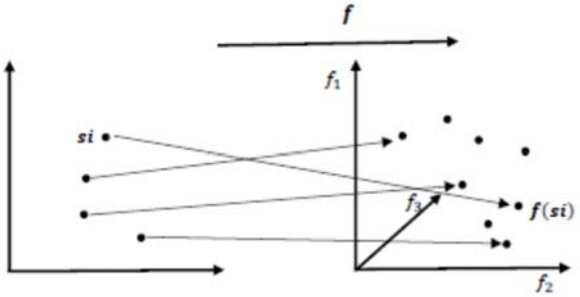

Fig. 1 Decision space mapping onto objective space

III. System Models

A. Application and Architecture

The two main parts of an embedded real-time system are the application and the architecture parts. In this work, the application is modelled by a data flow graph. Each vertex is a task and each edge a data dependency. Data dependencies express precedence constraints of the tasks and they have to be pertinently satisfied. Architecture model is represented by an undirected complete graph where each vertex is a processor, and each edge is a communication link. Processors are considered as heterogeneous and the communication mechanism as the send/receive one.

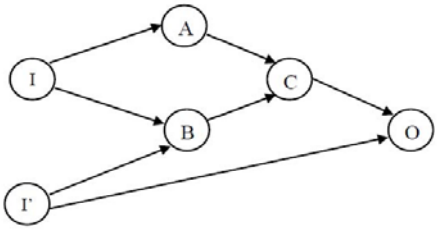

Fig. 2 Example of application composed of 6 tasks

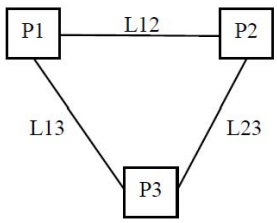

Fig. 3 Example of architecture with 3 processors

B. Reliability and Makespan Models

According to the model proposed in [8] 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 as described in (3):

$$Rel = e^{-\lambda d} \tag{3}$$

In [8], it is noted that some technical difficulties raise when using both reliability and makespan as objectives since the reliability depends intrinsically on the duration of the tasks and communications. Consequently, instead of using the usual model of the reliability [9], the concept of GSFR (Global System Failure Rate) proposed in [8] and noted ∧ is used. The GSFR expressed by (4) is the failure rate per time of the obtained multi-processors schedule, it is noted $\wedge(s)$ and defined as:

$$\wedge(s) - \frac{\log Rel(S)}{U(S)} \tag{4}$$

where $U(S)$ is the total utilization of the hardware resources and Rel the corresponding reliability. That signifies the system is seen as a single operation executed on one machine. Thus, the failure rate does not depend on the duration of the operation anymore. Otherwise, the makespan is the end execution time of the task that is completed last among all tasks. It is defined as in (5):

$$\mathrm{M} = \max_{pj}\left\{\max_{ti\ on\ pj} end\ (ti, pj)\right\} \tag{5}$$

where, end (t_i,p_j) is the time at which task t_i ends its execution on processor p_j.

In [10], a function called schedule pressure is deduced from the application graph and is defined for each task $t_i \in T_{cand}^{(n)}$ (n referring to the heuristic step and cand to the set of candidate tasks meaning those not yet scheduled and whose predecessors are already scheduled) and each processor p_j. It is noted $\sigma^{(n)}$ and defined in (6):

$$\sigma^{(n)}(t_i, p_j) = S_{(t_i,p_j)}^{(n)} + \bar{S}_{t_i}^{(n)} - R^{(n-1)} \tag{6}$$

where $R^{(n-1)}$ is the critical path length of the partial schedule constituted of already scheduled tasks; $S_{(t_i,p_j)}^{(n)}$ is the earliest time at which the task t_i can start its execution on the processor p_j; $\bar{S}_{t_i}^{(n)}$ is the latest time from the end of t_i, defined to be the length of the longest path from t_i to output tasks of the application graph.

The schedule pressure is used to select the best task which minimizes the length of the critical path by introducing a priority between the tasks to be scheduled.

C. Execution Models

An execution time is defined for each pair (t_i, p_j), it represents the worst-case execution time of the task t_i on the processor p_j. Since processors are heterogeneous, a task could have different execution times on different processors due to the heterogeneity of these ones. Furthermore, to each pair (d_i,l_j) corresponds a time expressing the worst-case transmission or communication time of the data dependency d_i on the communication link l_j. The intra-processor communication time is supposed to be 0-time unit.

IV. The Proposed Approach

In this work, both quality and diversity of produced solutions

are taken into account. The alternate treatment of the objectives leads to solutions approaching optimal ones while a self-organization strategy consists into solution reorganization in order to improve their diversity.

A. Approach Principle

The proposed approach called GSFR-Makespan Compromise Algorithm (GMCA) is based on three modules: Two heuristics constrained by a GSFR value and a makespan value respectively and which work in an alternate manner to optimize the GSFR and the makespan. The third module is a self-organization strategy to better explore solution space.

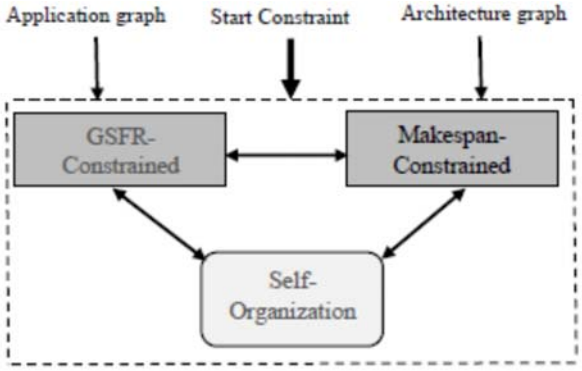

Fig. 4 The proposed approach

The self-organization is based on the neighborhood structure where a permutation-based neighborhood is used; it is defined by the transformation depicted in (7):

$$V: S \rightarrow P(s)\ such\ that: \forall\ s\ \in S \tag{7}$$

$V(s) = \{\ s_p\ /s_p$ is a schedule resulting from a given permutation p of $s\ \}$

Supposing that two solutions s and s' (grey and white colors), in the objective space, are too close, the self-organization consists into extending the decision space by applying permutations on s.

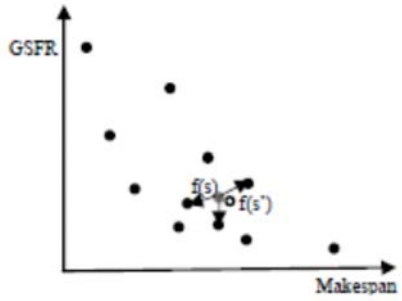

Fig. 5 Example of permutation result

The solution s could be replaced by one of its neighbours in the objective space. The starting constraint value could be changed using either GSFR or makespan value. This allows to create an instance of the problem.

B. The Proposed Heuristics

The proposed approach is supported by two greedy list-based heuristics (Algorithm 1 and Algorithm 2).

Algorithm 1: GSFR-Constrained Heuristic
Inputs: application graph, architecture graph, $GSFR$ constraint
Output: (GSFR value, Makespan value)
Begin
Initialize the lists of candidate and scheduled tasks: $T_{cand}^{(0)} := \{t \in T\ /\ pred(t) = \emptyset\}$
$T_{sched}^{(0)} := \emptyset$
While $T_{cand}^{(n)} \neq \emptyset$ **do**
1.Compute the *schedule pressure* for each task t_i of $T_{cand}^{(n)}$ on each processor pj such that GSFR value $\leq GSFR$ constraint;
2.Schedule the pair (candidate task *t*, processor *p*) such that *the schedule pressure* value is minimal;
3.Update the lists of candidate and scheduled tasks:
$T_{sched}^{(n)} := T_{sched}^{(n-1)} \cup \{t\}$

$T_{comp}^{(n+1)} := T_{comp}^{(n)} - \{t\} \cup \{t' \in succ(t)/pred(t') \subseteq T_{sched}^{(n)}\}$
end while
f(s) = (GSFR (s), Makespan (s)
if f(s) is too close to already produced compromise values **then**
execute Self-organization;
end if
end

GSFR and makespan objectives are optimized in an alternate manner while a self-organization is achieved to better explore the decision space and aiming to improve solution diversity.

The lists $T_{cand}^{(0)}$, $T_{sched}^{(0)}$ of candidate task list (a task is said to be candidate if it has no predecessor) and scheduled task list respectively are used. The two heuristics are guided by the same logic. Indeed, as long as there are still candidate (unscheduled) tasks, the three following steps are repeated:

1. For each candidate task, the cost-function value is calculated on each processor such that the specified constraint is satisfied.
2. The best pair (task, processor) is selected meaning the one minimizing the cost-function value and not violating the space defined by the current constraint value.
3. The two used lists are updated by adding the selected task to the schedule in construction and removing it from the list of candidate tasks.

In the case of close solutions, a self-organization is executed to eventually translate the current solution.

Algorithm 2: Makespan-Constrained Heuristic
Inputs: application graph, architecture graph, *Makespan* constraint
Output: (GSFR value, Makespan value)
Begin
Initialize the lists of candidate and scheduled tasks:
$T_{comp}^{(0)} := \{t \in T\ /\ pred(t) = \emptyset\}$
$T_{sched}^{(0)} := \emptyset$
While $T_{cand}^{(n)} \neq \emptyset$ **do**
1.Compute the GSFR for each task of $T_{cand}^{(n)}$ on each processor pj such that Makespan value $\leq$ *Makespan* constraint

2. Schedule the pair (candidate task *t*, processor *p*) such that GSFR value is minimal;
3. Update the lists of candidate and scheduled tasks:

$T_{sched}^{(n)} := T_{sched}^{(n-1)} \cup \{t\}$

$T_{comp}^{(n+1)} := T_{comp}^{(n)} - \{t\} \cup \{t' \in succ(t) / pred(t') \subseteq T_{sched}^{(n)}\}$

end while

$f(s) = (\text{GSFR}(s), \text{Makespan}(s))$

if f(s) is too close to already produced compromise values **then** execute Self-organization;

end if

end

The self-organization is described by Algorithm 3.

Algorithm 3: Permutation-based Self-Organization
Input: current schedule s, selected schedules
Output: New schedule s_N
Begin
Apply permutations on s to create a set of schedules;
Select the subset P_s of schedules satisfying precedence constraints;
if ∃ a schedule $s_k \in P_s$ such that
(1) s_k is not yet selected
and
(2) $f(s_k)$ is not close to compromise values of selected schedules
and
(3) $f(s_k)$ is non-dominated by the compromise values of schedules ∈ P_s
then
s_k is considered as the new schedule s_N
else
s is saved as the new schedule s_N
end if
end

When applying permutations on a given schedule, new schedules are produced. Once a solution (schedule) satisfies the three conditions cited in Algorithm 3, it is considered as the current schedule. However, such a schedule may not be found, in which case current schedule is saved.

V. Experiment Results

In order to evaluate the proposed approach, a comparison to SPEA2 algorithm [11] is realized. For this purpose, a set of random algorithm graphs and an architecture graph composed of 4, 5, and 6 processors is generated. The parameter to vary is the number of task N = 20, 40, 60, 80 and for each N, 100 graphs have been generated.

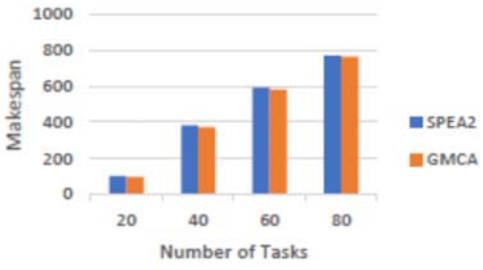

Fig. 6 Impact of N on Makespan for P = 5

The aim of simulations is to study the impact of N (number of tasks) and P (number of processors) on the reliability and the makespan.

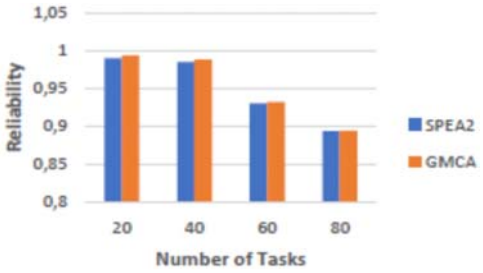

Fig. 7 Impact of N on Reliability for P = 5

Figs. 6 and 7 describe the impact of the number of tasks on the makespan and the reliability respectively. In Fig. 6, it is noted that GMCA performs better than SPEA2 since the makespan values related to GMCA are smaller than the ones of SPEA2. Similarly, with Fig. 7, it is observed that reliability values are greater when applying GMCA.

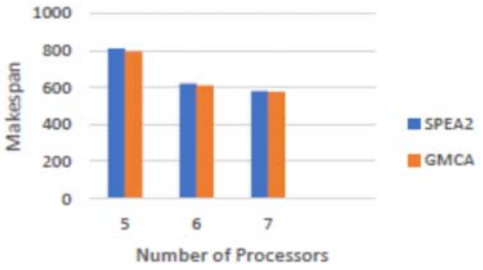

Fig. 8 Impact of P on Makespan for N = 40

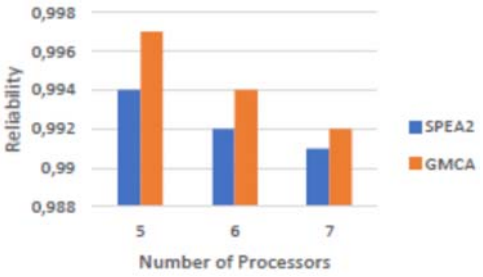

Fig. 9 Impact of P on Reliability for N = 40

TABLE I
Δ Metric Values of GMCA and SPEA2

GMCA Δ Metric	SPEA2 Δ Metric	GMCA Δ Metric	SPEA2 Δ Metric
0.535	0.539	0.632	0.639
0.605	0.611	0.540	0.543
0.712	0.720	0.582	0.584
0.615	0.630	0.701	0.705
0.541	0.542	0.652	0.655
0.630	0.641	0.679	0.681
0.553	0.556	0.714	0.718
0.701	0.710	0.579	0.583
0.663	0.670	0.597	0.601
0.529	0.532	0.705	0.710

Figs. 8 and 9 are about the impact of the number of processors on the makespan and the reliability. Fig. 8 shows that the makespan, when applying GMCA, is less so better than the ones produced by SPEA2 while in Fig. 9, the reliability is better with GMCA.

Solution diversity is evaluated using the Δ metric proposed in [12] where a smaller value of Δ implies a greater diversity. Table I shows more solution diversification with the proposed approach.

VI. Conclusion

In this paper, a self-organization-based approach for real-time systems design is presented. More precisely, the addressed problem is the mapping of an application composed of a set of tasks onto an architecture of heterogeneous processors. In addition, two conflicting objectives have to be optimized namely the makespan and the reliability. The proposed approach addresses both quality and diversity of solutions. Indeed, alternate treatment of the objectives aims to produce solutions of good quality whereas diversification is ensured by a self-organization strategy. This one consists into applying permutations over the decision space in order to reorganize solutions in a better diversified configuration. The approach evaluation is achieved through its comparison with SPEA2 algorithm. The obtained results show that the proposed approach produces solution of better quality with comparison to SPEA2. Furthermore, the Δ metric indicates that solution diversity is better when applying the proposed approach.

References

[1] G. Taheri, A. Khonsari, R. Entezari-Maleki, L. Sousa, "A hybrid algorithm for task scheduling on heterogeneous multiprocessor embedded systems," *Journal of Applied Soft Computing*, vol. 91, June. 2020

[2] R. Ramezani, Y. Sedaghat, M. Naghibzadeh, J. A. Clemente, "Reliability and Makespan Optimization of Hardware Task Graphs in Partially Reconfigurable Platforms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, pp. 983-994, 2017.

[3] A. K. Shukla, R. Nath, P. K. Muhuri, Q. M. Danish Lohani, "Energy efficient multi-objective scheduling of tasks with interval type-2 fuzzy timing constraints in an Industry 4.0 ecosystem," *Journal of Engineering Applications of Artificial Intelligence*, vol. 87, January. 2020.

[4] S. Shanthaveeraiah, Harsha, R. Balsubramani, "Optimal Power Management System in Embedded Devices by using Novel Scheduling Algorithm," *International journal of Embedded Systems and Real Time Communication Systems,* vol. 11, pp. 41-61, mois. 2020.

[5] M. Salimi, A. Majd, L. Loni, T. Seceleanu, C. Seceleanu, M. Sirjani, M. Daneshtalab, E. Troubytsina, "A Multi-objective Task Scheduling Method for Embedded System Design," in *Proc. 6th Conf. Engineering of Computer Based Systems*, 2020, pp. 1-9.

[6] M. Sehrish, A. Shabir, U. Israr, P. Dong Hwan, K. DoHyeun, "An Adaptive Emergency First Intelligent Scheduling Algorithm for Efficient Task Management and Scheduling in Hybrid of Hard Real-Time and Soft Real-Time Embedded IoT Systems," Sustainability, MDPI, Open Access Journal, vol. 11(8), April. 2019.

[7] S. S. Bendib, H. kalla, S. Kalla, R. Hocine "A Self-Organized Scheduling Algorithm for Embedded Real-Time Systems," *International journal of Embedded of Real-Time Communication Systems*, to be published.

[8] A. Girault, H. Kalla, "A novel bicriteria scheduling heuristics providing a guaranteed global failure rate," *IEEE Transactions on Dependable and secure Computing*, vol. 6, pp. 241-254, 2009.

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

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

[11] E. Zitzler, M. Laumanns, L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multi-objective Optimization," *Int. Conf. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*. 2001.

[12] K. Deb, L. Thiele, M. Laumanns,, E. Zitzler,"Scalable test problems for evolutionary multi-objective optimization, " *Evolutionary Multi-objective Optimization*, pp. 105-145, 2005.