A Hybrid Genetic based Approach for Real-time Reconfigurable Scheduling of OS Tasks in Uniprocessor Embedded Systems

Ibrahim Gharbi[1,2], Hamza Gharsellaoui[3,4,5] and Sadok Bouamama[1]
[1]*National School of Computer Science, ENSI, University of Manouba, Manouba, Tunisia*
[2]*Arar College of Technology, TVTC, Al Hudud ash Shamali, Kingdom of Saudi Arabia (KSA)*
[3]*Higher School of Technology and Computer Science of Tunis (ENICarthage), Carthage University, Carthage, Tunisia*
[4]*LISI-INSAT Laboratory, (INSAT), Carthage University, Carthage, Tunisia*
[5]*Al-Jouf College of Technology, TVTC, Al-Jouf, Kingdom of Saudi Arabia (KSA)*

Keywords: Real-time Scheduling, Genetic Algorithms, Reconfigurable Embedded Systems, Hybridization.

Abstract: This paper deals with the problem of scheduling uniprocessor real-time tasks by a hybrid genetic based scheduling algorithm. Nevertheless, when such a scenario is applied to save the system at the occurrence of hardware-software faults, or to improve its performance, some real-time properties can be violated at run-time. We propose a hybrid genetic based scheduling approach that automatically checks the systems feasibility after any reconfiguration scenario was applied on an embedded system. Indeed, if the system is unfeasible, the proposed approach operates directly in a highly dynamic and unpredictable environment and improves a rescheduling performance. This proposed approach which is based on a genetic algorithm (GA) combined with a tabu search (TS) algorithm is implemented which can find an optimized scheduling strategy to reschedule the embedded system after any system disturbance was happened. We mean by a system disturbance any automatic reconfiguration which is assumed to be applied at run-time: Addition-Removal of tasks or just modifications of their temporal parameters: WCET and/or deadlines. An example used as a benchmark is given, and the experimental results demonstrate the effectiveness of proposed genetic based scheduling approach over others such as a classical genetic algorithm approach.

1 INTRODUCTION

Today, real-time embedded systems are found in many diverse application areas including; automotive electronics, avionics, telecommunications, space systems, medical imaging, and consumer electronics. In all of these areas, there is rapid technological progress. The software engineering principles for embedded system should address specific constraints such as hard timing constraints, limited memory and power use, predefined hardware platform technology, and hardware costs. On the other hand, the new generations of embedded control systems are addressing new criteria such as flexibility and agility (HGS12). For these reasons, there is a need to develop tools, methodologies in embedded software engineering and dynamic reconfigurable embedded control systems as an independent discipline. Each system is a subset of n tasks. Each task is characterized by its worst case execution times (WCETs) C_i, an offset (release time) a_i, a period T_i and a deadline D_i. The general goal of this paper is to be reassured that the system is feasible and meets real-time constraints even if we change its implementation and to correctly allow the minimization of the response time of this system after any reconfiguration scenario (HGS12). To obtain this optimization (minimization of response time), we propose a hybrid genetic-based approach in which a software agent is deployed to dynamically adapt the system to its environment by applying reconfiguration scenarios. A reconfiguration scenario means the addition, removal or update of tasks in order to save the whole system on the occurrence of hardware/software faults, or also to improve its performance when random disturbances happen at run-time. Indeed, many real-time systems rely on the EDF scheduling algorithm in the case of uni-processor scheduling theory. This algorithm has been shown to be optimal under many different conditions. For example, for independent, preemptive tasks, on a uni-processor, EDF is optimal in the sense that if any algorithm can find a schedule where all tasks meet their deadlines, then

Gharbi I., Gharsellaoui H. and Bouamama S..
A Hybrid Genetic based Approach for Real-time Reconfigurable Scheduling of OS Tasks in Uniprocessor Embedded Systems.
DOI: 10.5220/0005463903850390
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 385-390
ISBN: 978-989-758-096-3
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

EDF can meet the deadlines (Der74).

According to (BV11), a hyper-period is defined as $HP = [\zeta, 2*LCM+\zeta]$, where LCM is the well-known Least Common Multiple of the tasks periods and ζ is the largest task offset. This algorithm, in our original paper assumes that sporadic and aperiodic tasks span no more than one hyper-period of the periodic tasks HP and the minimization of the response time is evaluated for each reconfiguration scenario.

The organization of the paper is as follows. Section 2 introduces the related work of the proposed approach. In Section 3, we present the new genetic-based approach for hybrid optimal scheduling theory. Section 4 presents the performance study and discusses experimental results of the proposed approach research. Section 5 summarizes the main results and presents the conclusion of the proposed approach and describes the intended future works.

2 BACKGROUND

We present related works dealing with reconfigurations and real-time scheduling of embedded systems. Companies building embedded real-time systems are driven by a profit motive. To succeed, they aim to meet the needs and desires of their customers by providing systems that are more capable, more flexible, and more effective than their competition, and by bringing these systems to market earlier. This desire for technological progress has resulted in a rapid increase in both software complexity and the processing demands placed on the underlying hardware (BV11).

Nowadays, several interesting studies have been published to develop reconfigurable embedded control systems. In (CAM05) Marian et al. propose a static reconfiguration technique for the reuse of tasks that implement a broad range of systems. The work in (MNRE07) proposes a methodology based on the human intervention to dynamically reconfigure tasks of considered systems. In (ASV07), an ontology-based agent is proposed by Vyatkin et al. to perform system reconfigurations according to user requirements and also the environment evolution. Window-constrained scheduling is proposed in (WS99), which is based on an algorithm named dynamic window-constrained scheduling (DWCS). In (PBC02), a window-constrained execution time can be assumed for reconfigurable tasks in n among m windows of jobs. In the following, we consider periodic, sporadic and aperiodic tasks. So, we note that the optimal scheduling algorithm based on the EDF principles and on the dynamic reconfiguration scenario is that we propose in the current original work in which we give solutions computed and presented to respond to the systems requirements.

3 GENETIC ALGORITHMS APPROACH

Based on the (BLS06) works, we will describe our hybrid approach in order to achieve our objective. We assume that our system is a mixture of a set of n periodic, sporadic and aperiodic tasks noted ξ^n. By considering that m new tasks are added to ξ^n, our system becomes ξ^{n+m} which is composed of n old tasks and m new tasks and some old/new tasks can miss their deadlines. At this moment, a reconfiguration scenario is automatically applied at real-time to adapt the system to its environment. Or to have a feasible system after each reconfiguration scenario, the difference between the deadline and the end of execution of each task must be greater or equal to 0.

3.1 Model

In this section we will present some preliminaries concepts and we will describe our contribution after.
ξ denotes a set of active sporadic and aperiodic tasks σ_i ordered by increasing deadline in a linked list.
a_i denotes the arrival time of task σ_i.
C_i denotes the maximum computation time of task σ_i.
c_i denotes the dynamic computation time of task σ_i, i.e., the remaining worst case execution time needed for the processor, at the current time, to complete task $\sigma_{i,k}$ without interruption.
d_i denotes the absolute deadline of task τ_i.
D_i denotes the relative deadline of task σ_i. S_i denotes the first start time of task σ_i. s_i denotes the last start time of task σ_i.
f_i denotes the estimated finishing time of task σ_i.
L_i denotes the laxity of task σ_i.
R_i denotes the residual time of task σ_i. Baruah et al. (SBS91) present a necessary and sufficient feasibility test for synchronous systems with pseudo-polynomial complexity. For this reason, we present the following relationships among the parameters defined above:
$d_i = a_i + D_i$ (1)
$L_i = d_i - a_i - C_i$ (2)
$R_i = d_i - f_i$ (3)
$f_1 = t + c_1$; $f_i = f_{i-1} + c_i \ \forall\ i > 1$ (4)
$R_1 = d_1 - t - c_1$ (5)
$R_i = R_{i-1} + (d_i - d_{i-1}) - c_i$. (6) (BS93)
For any other task σ_i, with $i > 1$,
$f_i = f_{i-1} + c_i$ (7) and, by equation (3), we have:
$R_i = d_i - f_i = d_i - f_{i-1} - c_i = d_i - (d_{i-1} - R_{i-1}) - c_i = R_{i-1} + (d_i - d_{i-1}) - c_i$

3.2 Chromosomes

In our work, we encode the solution to the problem into a chromosome to specify a GA and as consequence, we define/implement the crossover operator and the tabu search algorithm against the mutation operator on the works of authors in (BLS06).

To optimize the problem, we will optimize the arrival times of all sporadic and aperiodic tasks of all the genes composing the chromosome. A gene can be depicted as a pair of character and integer values (τ_i, $a_{i,j}$); that is a sporadic or aperiodic task symbol and an arrival time. The chromosome's size which is the number of genes was defined as the total number of executions of all sporadic and aperiodic tasks. The length of the chromosome is l = $\sum_{i=1}^{h} \left\lceil \frac{H}{min_i} \right\rceil$, where h is the number of sporadic and aperiodic tasks and k_i = $\left\lceil \frac{H}{min_i} \right\rceil$ is the maximum number of executions for sporadic/aperiodic task τ_i over an hyper-period H. As defined by (BLS06), to depict a non-existent arrival time, we use a special value: -1. To handle the constraints of the genes of the chromosome, we need a good design. These constraints are defined by two consecutive arrival times for a particular event which must have a difference of at least the minimum inter-arrival time, and at most the maximum inter-arrival time if it exists, else, it is set to H. Also, in order to facilitate chromosome manipulations, all genes corresponding to the same task τ_i are grouped together and ordered increasingly in a linked list $Link_i$ according to $a_{i,j}$. For example, given a set of 3 aperiodic tasks τ_1 (min_{τ_1} = 6); τ_2 (min_{τ_2} = 9) and τ_3 (min_{τ_3} = 12). We consider the following chromosome (τ_1 , -1) (τ_1 , 10) (τ_1 , 16) (τ_2 , 11) (τ_2 , 20) (τ_3 , 12) (τ_3 , 25) (τ_3 , 37) as a valid chromosome in a time interval (hyper-period) $H = [\zeta, 2 * LCM + \zeta]$; with ζ = 0 and lcm of (6, 9, 12) = 36, i.e., H = $[0, 72]$. In contrast, the following chromosome (τ_1 , -1) (τ_1 , 10) (τ_1 , 16) (τ_2 , 11) (τ_2 , 20) (τ_3 , 12) (τ_3 , 25) (τ_3 , 31) is not valid since the minimum inter-arrival time of the third task is not satisfied by the two last genes.

3.3 Initialization

We randomly generate the initial population at first with respect to the constraints mentioned above. After that, we find the hyper-period H based on the lcm of all periods of tasks τ_i. The authors in (BLS06) used a value T as a maximum of the time interval but we use in our work the hyper-period H. The value of $a_{i,j}$ is randomly selected from a range determined by the arrival time of $a_{i,j-1}$ as well as min_i and max_i. If max_i is not specified, its value is set to H: $[a_{i,j-1} + min_i, a_{i,j-1} + max_i]$. Also, if there is no previous gene (i.e., j = 1), or the previous gene depicts a non-existent arrival time (i.e., $a_{i,j-1}$ = -1), the range is $[0, max_i]$. After that, the genes are ordered in an increasing order and putted in a linked list ($Link_i$).

3.4 Cross-over

Cross-over and mutation operators are the ways GAs explore a solution space. Hence, they must be formulated in such a way that they efficiently and exhaustively explore the solution space (LE98).

On the other hand, task of formulating operators is rather difficult as genetic operators must maintain allowability. In other words, genetic operators must be designed in such a way that if a constraint is not violated by the parents, it will not be violated by the children resulting from the operations (EEZ94).

In (BLS06), the authors adopt the cross-over and mutation operators. In contrast, in our original work, we will adopt cross-over operator and tabu search algorithm on behalf of the mutation operator and we will test this hybrid approach in order to prove its efficiency. Indeed, the crossover operator is special in order to respect the constraint that every task which is copied didn't repeated another time in order to respect the tasks unicity and creates two solutions or childs by combining two parents (HHS14).

The principle of the crossover operator is to pass traits from the parents to the two resulting childs (offsprings) with variety methods. In (M.A), the authors used the operator crossover called sexual crossover or n-point crossover. The general idea in n-point crossover adopted by (M.A) is that the two parent chromosomes are aligned and cut into n+1 fragments at the same places to identify the division points. After that, two new childrens are created by altering the genes of the parents. In (BLS06), the authors adopted the method of creating two new children by inheriting fragments from parents with a 50 % probability where the division points of the parents depend on K_i for each task τ_i. For more details, see (BLS06).

In our work, we adopt another method in order to have a better solution from generation to another. Indeed, for each pair of parents (chromosomes), two crossover integers i_1 and i_2 are randomly generated with the condition $1 \leq i_1, i_2 \prec$ h where h is the number of genes composing the parent chromosome, being scheduled. In the random position i_1, the first children (child1) inherits i_2 genes from parent$_1$ (P_1) and the second children (child2) inherits i_2 genes from parent$_2$ (P_2). Then, child1 inherits the remaining genes of P_2 and similarly, child2 inherits the remaining genes of P_1. Finally, the genes must be ordered in the two new offsprings (childs) in order to avoid violating constraints.

Now, let us consider the same example of three aperiodic tasks used by (BLS06) to more explain our crossover operator principle. Tasks are τ_1 (min_{τ_1} = 100), τ_2 (min_{τ_2} = 150) and τ_3 (min_{τ_3} = 200) like described in table 1. In this example, each parent was composed of five genes. The random running of the two integers i_1 and i_2 was done and we have i_1 = 2 and i_2 = 3 with the condition 1≤ i_1, $i_2 \prec$ 5. Then, at the position 2, the crossover operator copy 3 genes (2, 3, 4) from the parent 1 for child1 and and 3 genes (2, 3, 4) from the parent 2 for child2. Then, child1 inherits the first and the fifth genes from parent 2, and the second children child2 inherits the first and the fifth genes from parent 1. Table 1 shows an example of a chromosome composed of five genes.

Table 1: Crossover Operator.

Chromosome Name	Task τ_1	Task τ_2	Task τ_3
Parent$_1$ (P_1)	(τ_1, 25) (τ_1, 150)	(τ_2, -1) (τ_2, 150)	(τ_3, 0)
Parent$_2$ (P_2)	(τ_1, 5) (τ_1, 200)	(τ_2, 50) (τ_2, 200)	(τ_3, 55)
Children$_1$ ($Child_1$)	(τ_1, 5) (τ_1, 150)	(τ_2, -1) (τ_2, 150)	(τ_3, 55)
Children$_2$ ($Child_2$)	(τ_1, 25) (τ_1, 200)	(τ_2, 50) (τ_2, 200)	(τ_3, 0)

3.5 Tabu Search Algorithm

In GAs, there is a mutation operator which is occurred when a gene is randomly chosen and mutated, and the resulting chromosome is evaluated for its new fitness function value. In (BLS06), the authors define a mutation operator as an operator that mutates genes in a chromosome by altering their arrival times. This work is limited in the sense that every mutated gene must be inserted in the best place and insertions occur from the left to right along the executions of a task and if no suitable insertion location is found, then no task execution can be added among the already existing task execution and this gene must be rejected. This result is strong in time and space and especially in worst case when many genes are rejected.

In our work, we will replace the mutation operator by the tabu search (TS) algorithm like below: The tabu search solves the problem and returns the best solution found. Indeed, this algorithm replaced the mutation operator and evaluated in randomly generated chromosomes and returned a good results which proved our approach. To more explain this main idea of this TS algorithm, let us consider a chromosome represented by a set of n genes. We adopt the change of one gene by another of the same task as an (filled up) elementary transformation and we evaluated the Fitness-Function (fit) for the chromosome under study after each alteration of the selected gene. In fact, we select a gene of the chromosome and we will be interested by the genes before and after it. For each gene of its neighbors (gene before and gene after), we alter its arrival time. A new arrival time $a^{'}_{i,j}$ is chosen for it from the range $[a_{i,j-1}+min_i, a_{i,j-1}+max_i]$ by the dichotomic algorithm to reduce the search space and we evaluate the fitness function after each tabu iteration in the hyper-period. The same procedure was executed for the gene before if the selected gene is not the first one on the chromosome and for the gene after if the selected gene is not the last one on the chromosome also. After completion of this procedure, the new gene leads to the best fitness function in the current chromosome was chosen to replace the previous one and the worst individual characterized by the least fitness in the current chromosome will be replaced by the offspring (child) and this gene of the best solution is added to the tabu list.

Let us consider the following example, we assume that the gene 2 of the task τ_1 (child1) is randomly selected in the tabu search algorithm.

The chromosome is: (τ_1, 11) (τ_1, 61) (τ_1, 120) (τ_2, 3) (τ_2, 90) (τ_3, 0) and the selected gene for the tabu search procedure is the second gene of the first task, i.e., (τ_1, 61). The research space is composed by the first and the third genes, i.e., (τ_1, 11) and (τ_1, 120). For the gene (τ_1, 11), the range is $[0+50, 0+100]$ = $[50, 100]$ (min = 50 and max = 100).

For the gene (τ_1, 120), the range is $[61+50, 61+100]$ = $[111, 161]$. Now, we will explore the first solution space represented by the first range and the second one by the dichotomic research method in order to get the best value that optimizes the fitness function. The new value $a^{'}_{i,j}$ will replace the previous execution time on the gene (τ_1, 61). After calculate the new value is equal to 70 and we have the new gene (τ_1, 70) and the chromosome becomes (τ_1, 11) (τ_1, 70) (τ_1, 120) (τ_2, 3) (τ_2, 90) (τ_3, 0). This procedure is repeated until the changes leads to a valid chromosome.

3.6 Fitness Function

In our work based on a genetic algorithm, the evaluation of a model is obligatory. This evaluation is represented by a fitness function (objective function) that we noted fit. Indeed, fit(ch(τ)) = $\sum_{j=1}^{K_\tau} R_\tau$, where $R_\tau = d_{\tau,j} - f_{\tau,j}$. By this equation we have fit(ch(τ)) = $\sum_{j=1}^{K_\tau} d_{\tau,j} - f_{\tau,j}$ for target task τ. By considering the difference between the deadline of an execution and the execution's actual completion, i.e., $d_{\tau,j} - f_{\tau,j}$. We are thus interested in all periodic, sporadic and aperiodic tasks in a task set $\xi_h = \sum_{i=1}^{h} \tau_i$ which is composed by h tasks among m + n tasks. So, we have fit(ξ_h) = $\sum_{i=1}^{h} fit(ch(\tau_i)) = \sum_{i=1}^{h}\sum_{j=1}^{K_\tau}(d_{\tau_{i,j}} - f_{\tau_{i,j}})$. Here our system will be feasible if $R_\tau = d_{\tau,j} - f_{\tau,j} \geq 0$ and the solution is optimal if this fitness function value converges on zero.

4 EXPERIMENTATION RESULTS AND DISCUSSIONS

This section is intended for presenting the simulations results, the discussions and assumptions about the presented work. The advantages and disadvantages about them are discussed.

4.1 Simulations Results

In a real-world optimization problem, there often exist multiple benchmarks to be compared with test results. In this section, we will restraint to the benchmarks presented by (DZ13). We present results obtained with a real-coded genetic algorithm with a population size of 20 and a generation number of 50 in order to check reliability of a solution and feasibility of a task set model. The proposed GA uses the crossover operator and the TS algorithm presented in the above section. First, the proposed GA begins by the parameters of tasks presented in the benchmarks and then iterates to converge on a small particular fitness function. When this fitness function value converges on zero, we terminate the research and we consider this solution as optimal. The following table 2 shows the simulations results presented by our approach.

Table 2: Simulations Results.

Approach Name	Worst	Average	Best
Hybrid Approach	$1.020*10^{-8}$	$1.017*10^{-8}$	$1.010*10^{-8}$
Classical Approach	$1.028*10^{-8}$	$1.090*10^{-8}$	$1.021*10^{-8}$

4.2 Discussion

We perform 1000 runs with different task sets of the benchmark values. The first row in table 2 shows the performance of the proposed hybrid GA. We observe that in all 1000 runs, the GA is able to find a solution near the optimal feasible solution (10^{-8}). The second row in table 2 shows the results of the classical GA, i.e., crossover and mutation operators presented by (BLS06).

Now, as we show in table 2, if we want to compare the first hybrid method and the second classical method, we observe the performance of our proposed solution.

We conclude that, our proposed GA can find the correct reliable near optimal solution compared with the classical GA. In contrast, the classical GA is faster than our approach due to lateness obtained by the TS algorithm when the number of populations is so high.

Finally, we can say that genetic based algorithms can be computationally expensive in time but efficient, important and practice to difficult feasibility-based optimization problems.

5 CONCLUSION AND FUTURE WORK

In this paper we have proposed a real-time hybrid genetic based scheduling approach for solving the embedded systems feasibility problem. A specific fitness function fit is designed to guide the search. Computational results show that our approach is very promising. The approach was tested and the results indicate that performance of our original algorithm is better than others. Comparison with classical genetic approach demonstrates the effectiveness of the proposed rescheduling approach. Since we were unable to find a counter example for which the approach fails, we conjecture that it always finds an optimal solutions.

We plan to further strengthen this work to include distributed embedded systems problems.

ACKNOWLEDGEMENTS

The authors thank anonymous reviewers whose comments clarified and enriched the contents of this paper.

REFERENCES

Y. Al-Safi and V. Vyatkin. *An ontology-based reconfiguration agent for intelligent mechatronic systems.* Int. Conf. Hol. Multi-Agent Syst. Manuf., vol. 4659, pp. 114-126, Regensburg, Germany, 4th edition, 2007.

Lionel C. Briand, Yvan Labiche, and Marwa Shousha. Using genetic algorithms for early schedulability analysis and stress testing in real-time systems. *Genetic Programming and Evolvable Machines*, 7(2):145–170, 2006.

G. Buttazzo and J. Stankovic. *RED: Robust Earliest Deadline Scheduling*. Int. Workshop On Responsive Computing Systems, Austin, 3rd edition, 1993.

Ballester R. Ripoll L. Brocal V., BalbastreP. *Task period selection to minimize hyperperiod, Emerging Technologies & Factory Automation (ETFA), IEEE conference on, pp.1-4, 2011*. doi: 10.1109/ETFA.2011.6059178, Toulouse, France, 16th edition, 2011.

K. Sierszecki C. Angelov and N. Marian. *Design models for reusable and reconfigurable state machines*. L.T. Yang et al., Eds., Proc. of Embedded Ubiquitous Comput., 2005.

M. Dertouzos. *Control Robotics: The Procedural Control of Physical Processes*. Proceedings of the IFIP Congress, 1974.

Stephen C. H. Leung Qingshan Chen Defu Zhang, Lijun Wei. A binary search heuristic algorithm based on randomized local search for the rectangular strip packing problem. *INFORMS Journal on Computing*, 25(2):332–345, 2013.

Eiben A. E., Raue P. E., and Ruttkay Z. Solving constraint satisfaction problems using genetic algorithms. In *IEEE World Conference on Evolutionary Computing*, pages 542–547, 1994.

M.Khalgui H. Gharsellaoui and S.BenAhmed. *Feasible Automatic Reconfigurations of Real-Time OS Tasks*. IGI-Global Knowledge, USA, isbn13: 9781466602946 edition, 2012.

Gharsellaoui H., Hasni H., and Ben Ahmed S. Real-time reconfigurable scheduling using genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO*, Vancouver, BC, Canada, 2014.

Haupt R. L. and Haupt S. E. *Practical Genetic Algorithms*. Wiley-Interscience, 1998.

M.A.Pawlowsky. Crossover operators. In *Practical Handbook of Genetic Algorithms Applications*.

T. Strasser A. Zoitl O. Hummer M. N. Rooker, C. Sunder and G. Ebenhofer. *Zero downtime reconfiguration of distributed automation systems: The CEDAC approach*. Int. Conf. Indust. Appl. Holonic Multi-Agent Syst., Regensburg, 3rd edition, 2007.

I. Ripoll P. Balbastre and A. Crespo. *Schedulability analysis of window-constrained execution time tasks for real-time control*. 14th Euromicro Conf. Real- Time Syst., 14th edition, 2002.

B. Mishra A. Raghunathan L. Rosier S. Baruah, G. Koren and D. Shasha. *On-line Scheduling in the Presence of Overload*. IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1991.

R. West and K. Schwan. *Dynamic window-constrained scheduling for multimedia applications*. IEEE 6th Int. Conf. Multi. Comput. Syst., 6th edition, 1999.