Combinatorial Approaches for Low-power and Real-time Adaptive Reconfigurable Embedded Systems

Hamza Chniter[1], Fethi Jarray[2] and Mohamed Khalgui[1]
[1]*INSAT Institute - University of Carthago, Tunis, Tunisia*
FST faculty - University of Elmanar, Tunis, Tunisia
[2]*ISI Institute, Mednine, Tunisia*

Keywords: Embedded System, Reconfiguration, Real-Time and Low-Power Scheduling, Mixed-Integer Linear Programming, Simulated Annealing.

Abstract: This paper describes an optimisation-oriented approach to dynamic reconfiguration of embedded systems with real-time and power consumption constraints. A reconfiguration scenario is assumed to be any run-time operation allowing the addition-removal-update of OS tasks to adapt the system to its environment under well-defined conditions. The problem is that any reconfiguration can lead the system to an unfeasible state where temporal properties are violated or the energy consumption is well-increased. Two methods, integer programming and simulated annealing, are used to that purpose. The methods have been validated using analysis tools to evaluate the whole contribution.

1 INTRODUCTION

Nowadays, embedded systems (ES) are integrated in larger architectures to interact continuously with their environment under functional and temporal constraints (GAUJAL and al, 2003). These ES generally include both software and hardware components which offer today many advantages like the run-time reconfiguration of the system. A reconfiguration is any operation allowing the adaptation of the system to its environment under well-defined constraints (Quadri and al, 2012). The embedded systems are characterized also by their dedicated function and the high requirements on reliability and correctness. In fact, it will be able to react with the environment and meet various functional and extra-functional (e.g. temporal) constraints. These systems run often under power constraints that can be violated after particular reconfiguration scenarios (addition of heavy OS tasks). Consequently, some processor technologies are used to make the trade-off between time execution, speed and energy consumption. One of the new technologies called DVS (Dynamic Voltage Scaling) (GRUIAN, 2002) is integrated into processors to dynamically change the execution speed of tasks. It aims to change the operating frequency of the processor during execution from a set of available speeds.

We assume in the paper a system of n synchronous periodic tasks which meet the corresponding deadlines to be described in user requirements. If tasks are asynchronous, we can transform them to synchronous. We assume a run-time reconfiguration scenario to add m new tasks in order to adapt the system to its environment under well-defined constraints. We assume that the new system of $(n+m)$ tasks is not feasible where some deadlines are violated and the energy consumption increases. The problem is how can we compute a new processor frequency in order to obtain a new feasible system after this reconfiguration? To solve the problem, we propose two combinatorial approaches: integer programming and simulated annealing. These two approaches have been used to minimize the energy consumption after any reconfiguration scenario by determining the suitable processor frequency to satisfy also the temporal properties. We prove in particular the performance of the integer programming to the simulated annealing. Some experimentations are applied at the end of the paper to evaluate the whole contribution.

The remainder of this paper is organized as follows. We discuss in Section 2 the originality of this paper by studying the state of the art. Section 3 exposes the problem. We present in Section 4 some terminologies and the contribution dealing with integer program formulation and a simulated annealing for reconfigurable embedded systems. Finally, numerical

Chniter H., Jarray F. and Khalgui M..
Combinatorial Approaches for Low-power and Real-time Adaptive Reconfigurable Embedded Systems.
DOI: 10.5220/0004752701510157
In *Proceedings of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems* (PECCS-2014), pages 151-157
ISBN: 978-989-758-000-0

results are presented and discussed in Section 5.

2 RELATED WORKS

The paper presents the scheduling problem of periodic non preemptive tasks that implement a reconfigurable embedded system while minimizing the energy consumption. There are many papers in the literature on algorithms providing approximating results for this NP-complete problem. In (Heilmann, 2003), the author presented an exact solution for the general resource-constrained project scheduling problem to determine a mode and a start time for each activity such that all constraints are respected and the duration of the project is minimized. The solution method is a depth-first search based branch&bound procedure. The work in (Xu, 1993) presented a branch and bound approach to solve allocation problem communicating periodic tasks. In another work (GAUJAL and al, 2003), Gauja presented an algorithm based on finding the shortest path in a graph to solve the same problem. In (Hladik and al, 2008), the authors presented Integer Linear Programming (ILP) for scheduling problem with dependent tasks in a multiprocessor homogeneous system. (Kuei-Tang and al, 2013) applied DVFS techniques to mobile computing platforms where performance constraints, such as task deadlines. They try the problem in a linear program and solve the problem by the simplex algorithm. The objective function is to make the trade-off between the total weighted tardiness and the power cost. A scheduling based on constraint programming multi-objective (multi-criteria optimization) is proposed in (Hladik and al, 2008). In (Majazi and Ghorbanali., 2012), the authors focused on the multi-objective flexible job-shop scheduling problem with parallel processors and maintenance cost. Two meta-heuristic algorithms, an hybrid genetic algorithm. Based on DVS technology, Jeannenot proposed in (Jeannenot and al,2004) a set of algorithms under periodic real-time tasks in a processor with dynamic variable speed. The authors seek to determine the suitable speeds execution for each task to minimize the total energy consumption from a real-time feasible embedded system.

Although the cited works are interesting and important, they do not address reconfigurable systems that can dynamically change their behaviors at run-time. We expose in the current paper the problem of reconfigurable systems by using Integer programming and simulated annealing. The goal is to compute the frequency processor and the execution sequence of tasks with a good performance in terms of energy cost and execution time.

3 PROBLEM AND NOTATION

We detail the problem in this section and present a terminology to be followed in the current paper. We assume a reconfigurable real-time system to be composed of periodic independent tasks that we assume synchronous. A reconfiguration scenario is any run-time operation allowing the addition-removal-update of tasks to adapt the system to its environment. Nevertheless, the application of a scenario can increase the energy consumption or push some tasks (new or old) to violate corresponding deadlines. Our goal is to provide some solutions that will optimize the energy consumption and guarantee the respect of deadlines after each reconfiguration scenario. We propose two approaches Integer Programming *IP* Model and Simulated Annealing *SA* to find the required solution by changing the processor speed. We want also to compare these two approaches to find the optimal and best solution. The integer programming approach is based on a mathematical model including the objective function and the constraints in relation, the simulated annealing heuristic is inspired from a process used in metallurgy, many parameters will be fixed to turn this heuristic and to give the expected solution such as the initial solution with which it must start, the initial temperature and the maximum number of iterations.

Notation:

We consider a set of n periodic tasks $T_i, i = 1 \dots n$. Each task i is classically characterized by four parameters. Firstly by its release (or arrival) time r_i, i.e each task T_i cannot begin execution before time r_i. Secondly by its absolute deadline constraint d_i, i.e. each task should finish before time d_i. Thirdly by its computation time at the normalized processor frequency C_{ni}. Finally by its period which is equal to the deadline (Liu and Layland, 1973).

We denote respectively by f_n and V_n the normalized frequency and voltage of the system. We assume that there're usually proportional. We suppose that each task T_i is executed at frequency F_i and at voltage V_i. We denote by η_i the reduction factor of voltage when T_i is executed, $V_i = \frac{V_n}{\eta_i}$. So $C_i = C_{ni}\eta_i$. In general, when the system is running at frequency F and voltage V, the power consumption is
$P = CV^2F$ where C is a constant related to the circuit type of processor (Chuan and al, 2012). If the system is running over x times, the energy consumption is: $E = Px$.

The problem is then to allow low-power and real-time optimal scheduling of reconfigurable tasks after each reconfiguration scenario.

The power P_i to be consumed by task T_i is :
$P_i = CV_i^2 F_i = C\frac{V_n f_n}{\eta_i^3}$.
The energy E_i consumed by the task T_i is:
$E_i = P_i C_i = C\frac{V_n f_n C_{ni}}{\eta_i^2} = K\frac{C_{ni}}{\eta_i^2}$ where the constant $K = CV_n F_n$.
So the total energy consumption of the system is:

$$E = \sum_{i=1}^{n} E_i = K\sum_{i=1}^{n}\frac{C_{ni}}{\eta_i^2} \tag{1}$$

As a running example to be followed in the next sections, we assume a real-time embedded system to be composed first of 5 tasks as depicted in table 1-page 3. We assume a run-time reconfiguration scenario to add 3 new OS tasks in order to adapt the system under particular constraints. The problem is then to guarantee the feasibility of these eight tasks after this scenario while satisfying also the energy constraints.

Table 1: Current system configuration

Tasks	Release time	WCET	deadline	period
T_1	0	13	80	80
T_2	0	6	70	70
T_3	0	39	90	90
T_4	0	13	110	110
T_5	0	26	100	100

The current system is feasible and was tested by the real-time scheduling simulator 'chaddar' (Singhoff and al, 2004). The energy consumption is equal to $2.328J = 2328mW$ and the CPU charge is equal to 1.059. The CPU charge factor U was calculated by equation (2) and the energy by equation (1) previously presented.

$$U = \sum_{i=1}^{n}\frac{C_i}{d_i}. \tag{2}$$

Where C_i, d_i are respectively the execution time and the deadline of task i and n denotes the number of tasks in the system. We assume a reconfiguration scenario by adding 3 additional tasks in table **??**-page 3 to the current system, so the new system becomes infeasible because we have some tasks that miss their deadlines(T_5, T_8, T_4) and the CPU charge increases to 1.427. The energy consumption is also increased and becomes $3,168J = 3168mW$.

We propose as a possible solution to modify the processor speed by using integer programming model or simulated annealing to meet the corresponding deadlines and to optimize the energy consumption.

Table 2: New system configuration.

Tasks	Release time	WCET	deadline	period
T_1	0	13	80	80
T_2	0	6	70	70
T_3	0	39	90	90
T_4	0	13	110	110
T_5	0	26	100	100
T_6	0	10	85	85
T_7	0	11	94	94
T_8	0	14	105	105

4 CONTRIBUTION: FEASIBLE RECONFIGURATION FOR EMBEDDED SYSTEM

We define in this section the two solutions that we propose for the modification of the processor speed after any reconfiguration scenario.

4.1 Integer Programming Model

We seek to minimize the total energy consumption of the system under various operating constraints, the energy is defined as above:

$$E = \sum_{i=1}^{n} E_i = K\sum_{i=1}^{n}\frac{C_{ni}}{\eta_i^2} \tag{3}$$

We introduce the starting time t_i to denote the effective starting time of task T_i. Our goal to minimize the total consumed energy under the following constraints:

a) No simultaneously executed tasks
To ensure a single executed task in a time, we should have either $t_j - t_i - C_i \geq 0$ or $t_i - t_j - C_j \geq 0$ or for every pair of tasks T_i and T_j . This condition can be rewritten as $t_i - t_j \geq C_{nj}\eta_j - M\alpha_{ij}$ and $t_j - t_i \geq C_{ni}\eta_i - M(1-\alpha_{ij})$ where α_{ij} is a binary variable and M is a big constant. $\alpha_{ij} = 1$ means that T_j is executed before T_i.

b) Deadline of each task should be respected

$$t_i + C_{ni}\eta_i \leq d_i \tag{4}$$

c) The the release time should be respected, $t_i \geq r_i \; \forall \; i$. Thus the basic model is the following

$$P\begin{cases} Minimize\ \ K\displaystyle\sum_{i=1}^{n}\frac{C_{ni}}{{\eta_i}^2} \\ s.t. \\ t_i - t_j \geq C_{nj}\eta_j - M\alpha_{ij} \\ t_j - t_i \geq C_{ni}\eta_i - M(1-\alpha_{ij}) \\ t_i + C_{ni}\eta_i \leq d_i\ \forall\ i \\ t_i \geq r_i\ \forall\ i \\ t_i \geq 0\ \forall\ i \\ \alpha_{ij} \in \{0,1\}\ \forall\ i<j \\ \eta_i \geq 0\ \forall\ i \end{cases} \qquad (5)$$

It is easy to incorporate into P the finish time of the system 'T' by adding the following constraint: $T \geq t_i + C_{ni}\eta_i\ \forall\ i$. The extended program is:

$$PE\begin{cases} Minimize\ \ K\displaystyle\sum_{i=1}^{n}\frac{C_{ni}}{{\eta_i}^2} + T \\ t_i - t_j \geq C_{nj}\eta_j - M\alpha_{ij} \\ t_j - t_i \geq C_{ni}\eta_i - M(1-\alpha_{ij}) \\ t_i + C_{ni}\eta_i \leq d_i\ \forall\ i \\ t_i \geq r_i\ \forall\ i \\ T \geq t_i + C_{ni}\eta_i\ \forall\ i \\ t_i \leq 0\ \forall\ i \\ \alpha_{ij} \in \{0,1\}\ \forall\ i<j \\ \eta_i \geq 0\ \forall i \end{cases} \qquad (6)$$

Since the objective function is fractional and the constraints are linear, its not possible to solve the program P or PE by any solver thus we simplify this program by maximizing the min of the reduction factor η_i. The non-linearity of the program PE is due to the fact that the computation time of tasks is proportional to the reduction factor whereas the energy consumption is inversely proportional to the reduction factors. The simplified program is:

$$PS\begin{cases} Maximize\ \ x \\ s.t. \\ t_i - t_j \geq C_{nj}\eta_j - M\alpha_{ij} \\ t_j - t_i \geq C_{ni}\eta_i - M(1-\alpha_{ij}) \\ t_i \leq M\ \forall\ i \\ t_i + C_{ni}\ \eta_i \leq d_i\ \forall\ i \\ t_i \geq r_i\ \forall\ i \\ x \leq \eta_i\ \forall\ i \\ 0 \leq \eta_i\ \forall\ i \\ x \leq \eta_i\ \forall\ i \\ t_i \geq 0\ \forall\ i \\ \alpha_{ij} \in \{0,1\}\ \forall\ i<j \\ C_{ni} \geq 0\ \forall\ i \end{cases} \qquad (7)$$

Case study:

Our model was applied to the system recently presented to resolve the non feasibility problem. It computes for each task, start time, finish time and new the WCET after changing the reduction factor of the processor speed. The results are presented in table 4.

Table 3: Applied model for WCET reconfiguration.

Tsks	Release time	Last WCET	New WCET	Reduction factor	Start time	Finish time	deadline
T_1	0.00	13.00	9.105	0.70	0.00	9.105	80
T_2	0.00	6.00	4.202	0.70	9.105	13.307	70
T_3	0.00	39.00	27.316	0.70	13.307	40.623	90
T_4	0.00	13.00	9.105	0.70	40.623	49.729	110
T_5	0.00	26.00	18.210	0.70	49.729	67.939	100
T_6	0.00	10.00	7.004	0.70	67.939	74.944	85
T_7	0.00	11.00	7.704	0.70	74.944	82.648	94
T_8	0.00	14.00	9.805	0.70	82.648	92.454	105

4.2 Simulated Annealing Approach

We have also proposed simulated annealing approach to proof its performance in a reconfigurable real-time embedded system and to compare it to integer programming and related works. The simulated annealing *SA* is based on neighborhood search. In local search techniques, we generally start with a random solution and try to improve it over the iterations. Greedy heuristics always move from the current solution to the best neighboring solution. In order to escape local minima, simulated annealing allows uphill moves in a controlled manner. At each step, we generate a perturbation, if the objective function decreased, then, we accept the new state, else we accept the new state with a probability related to this increase. We have to choose an initial temperature and to ensure the stop criteria.

Initial solution:

A solution should only respect the release times of the tasks, i.e. we relax the rest of constraints. The initial solution can be computed by the following way:

1. Sort the tasks according to their release times.
2. If two tasks have the same release times, we choose the one with the closest deadline.
3. Repeat the two steps until choosing all tasks.

Objective function:

The objective function is to minimize the sum of the total energy consumption, the total execution time and the number of tasks that miss their deadlines.

Neighborhood structure:

The neighborhood of a solution is a set of solutions that can be reached from the solution by a simple operation (move). Given the current solution, represented by the set of tasks, their starting times and effective frequencies, a neighbor solution is built by either swapping the execution order of two randomly selected tasks or changing the frequency of a random task beyond its deadlines.

Simulated annealing parameters:

The main parameters of the simulated annealing are the initial temperature, the temperature length, the cooling ratio and the stopping criterion. The initial temperature (90) must be high enough so that the final solution to be independent of the starting one. The temperature length (50) is the number of iterations at a given temperatures. However the temperature length may vary from temperature to temperature and it is important to spend long time at lower temperatures The cooling ratio ($\mu = 0.9$) is the rate at which the temperature is reduced. In our simulation, the simulated annealing stops when the minimum value of the temperature has been reached (5) or a certain number of temperatures (4) has passed without acceptance of a new solution or the number of total iterations (1000) has been executed.

5 NUMERICAL RESULTS

The integer programming model was solved with ILOG CPLEX 11.1 solver on a mono-processor core 2 duo , 1.2 Mhz and 1 Giga RAM. The simulated annealing was implemented in C and executed in the same machine.

Comparation with related work:

We note that the two approaches implemented in this paper to solve the problem of non feasibility of a reconfigurable real-time embedded system, give results close to those of (Jeannenot and al,2004). However the previous model developed does not guarantee the feasibility of the system. In addition our model allows to compute the feasibility more than the execution sequence of tasks but also the start and the finish time of each task. In our experimentation, we have randomly generated instances with 10 to 400 jobs. The numerical results are depicted in the table 4. The first column shows the size of the problem i.e the number of jobs. The sub-column labeled ”time” indicates the running time in seconds for each method. The sub-column labeled ”Energy” gives the total energy consumption. The sub-column labeled ”CPU charge” gives the total charge of the processor. The sub-column labeled ”Fitness” gives the objective function recently described. Finally, the sub-column labeled ”D exceeded” reveals the number of tasks that miss their deadlines.

Table 4 shows that the fitness function of the integer program is lower than that of the simulated annealing. The processor is less loaded in the integer program than in the simulated annealing. However for the large size instances, the simulated annealing is much faster. We conclude that the integer programming is more preferment than the simulated annealing algorithm for the small and the medium instances. Moreover integer program guarantees that all tasks are respected due to the constraint of deadlines (c).

Figures 1 and 2 present a graphic Comparation between simulated annealing and *IP* in term of CPU charge and energy consumption. In figure 1, *SA* gives a lower CPU charge than *IP* because the *IP* attempts to exploit up the processor to fulfill the needs of tasks and to meet all the deadlines. However, *SA* does not sometimes meet all the deadlines.

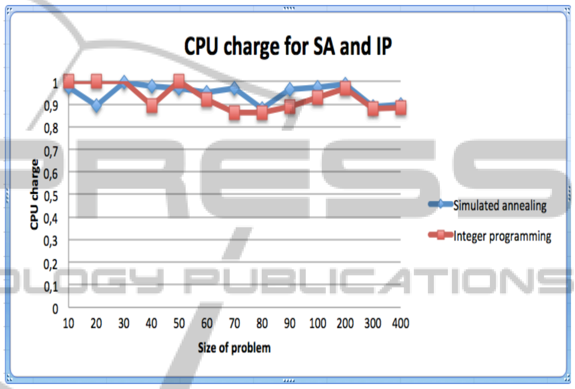

Figure 1: Comparation between *SA* and *IP* from CPU charge.

According to the energy consumption, we observe in figure 2 that *IP* is more effective as the number of instances increases because it allows to explore more the search space of solutions and can give a fairly optimal solution.

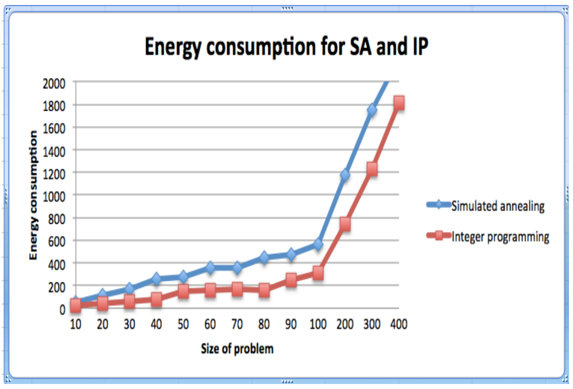

Figure 2: Comparation between *SA* and *IP* from energy consumption.

In figure 2, we compare the average CPU charge for the two proposed approaches and those presented as follows in (Jeannenot and al,2004) on instances of 5 to 15 tasks. Approaches marked with ’*’ correspond to the *SA* and the *IP* proposed in this paper, others

Table 4: Comparation between integer programing and simulated annealing.

	Simulated annealing					Integer programming				
basic system+added tasks	Time	Energy	CPU charge	Fitness	D exceeded	Time	Energy	CPU charge	Fitness	D exceeded
5+5	42ms	51.44	0,9743	64.04	1	2.52s	22.5	1	37.50	0
15+5	38ms	114.96	0,8929	134.76	1	8.03s	39	0,9999	64.99	0
20+10	50ms	169.96	0,99953	197.16	1	25.56s	55.5	1	92.50	0
30+10	45ms	255.61	0,9781	261.81	1	345.54s	75	0,8933	125.00	0
40+10	71ms	271.6	0,9716	315.19	0	1626.04s	147.75	1	215.99	0
45+15	69ms	353.43	0,9512	405.02	1	1623.3s	162.44	0,9215	247.03	0
50+20	84ms	358.06	0,9703	420.25	1	1624.68s	165.68	0,8629	259.18	0
60+20	52ms	448.67	0,8791	516.66	1	1621.42s	154.69	0,86452	252.75	0
70+20	67ms	469.64	0,9653	573.83	1	1621.63s	247.97	0,8913	251.69	0
80+20	68ms	561.13	0,9732	644.13	0	1708.29s	307.02	0,9283	453.98	0
150+50	278ms	1176.52	0,9892	1345.12	1	1880.57s	743,39	0,9711	513.05	0
250+50	659ms	1748.46	0,8908	2007.46	1	1950.89s	1230.76	0,8791	594.33	0
300+100	1s	2212.5	0,8996	2632.70	66	2226.04s	499.04	0,8839	753.61	0

refer to (Jeannenot and al,2004). Our approaches try to exploit the flexibility of the processor speed to meet the deadlines of tasks and to minimize the energy cost because in our contribution approaches will work in a reconfigurable real-time embedded system so that feasibility constraint after a reconfiguration scenario requires more resources of processors.

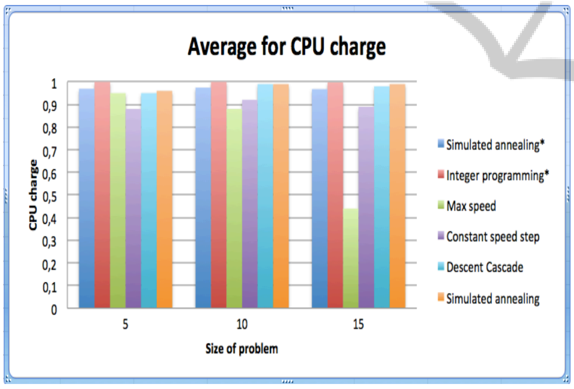

Figure 3: Average CPU charge from each approach.

6 CONCLUSIONS

The paper proposes two methods based on integer programming and simulated annealing to solve the non-feasibility scheduling problem while minimizing the energy consumption of a reconfigurable real-time embedded system. The numerical results show that the integer programming model gives good results and over performs the simulated annealing algorithm to resolve the problem. Both approaches give results close to those presented in (Jeannenot and al,2004) but they take advantage of the fact that They deal with reconfigurable real-time system and can ensure the system feasibility after any scenario. However, their effectiveness is not yet checked for other categories of tasks such as sporadic and aperiodic. The proposed model can be extended to include other constraints such as multiprocessor systems and other objectives such as minimization of the communication between the tasks.

REFERENCES

GAUJAL, B., AND NAVET, N. Ordonnancement sous contrainte de temps et d'énergie. In ACs de l'école d'été temps réel (ETR03) (9-12 Septembre2003), pp. 263 UTF2013276.

C. L. Liu, James W. Layland: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM 20(1): 46-61 (1973)

GRUIAN F., Energy-Centric Scheduling for Real-Time Systems, PhD thesis, Lund Institute of Technology, 2002.

GAUJAL, B., NAVET, N., AND WALSH, C. A linear algorithm for realtime scheduling with optimal energy use. INRIA Research report, 4886(July 2003).

Vahid Majazi Dalfard, Ghorbanali Mohammadi: Two meta-heuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel processor and maintenance constraints. Computers & Mathematics with Applications 64(6): 2111-2117 (2012).

Chuan He, Xiaomin Zhu, Hui Guo, Dishan Qiu, Jianqing Jiang: Rolling-horizon scheduling for energy constrained distributed real-time embedded systems. Journal of Systems and Software 85(4): 780-794 (2012).

Kuei-Tang Fang, Bertrand M. T. Lin: Parallel-processor scheduling to minimize tardiness penalty and power cost. Computers & Industrial Engineering 64(1): 224-234 (2013)

Roland Heilmann, A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. European Journal of Operational Research 144(2): 348-365 (2003).

RJ. Xu. Multiprocessor scheduling of processes with release times, deadlines, precedence, and exclusion relations. IEEE Transactions, on Parallel Distibuted Systems, 19(2), February 1993.

P-E. Hladik, H. Cambazard, A-M. Deplanche, and N. Jussien. Solving a real-time allocation problem with constraint programming. J. Syst. Softw., 81(1):132-149, 2008.

P-E. Hladik, H. Cambazard, A-M. Deplanche, and N. Jussien. Solving a real-time allocation problem with constraint programming. J. Syst. Softw., 81(1):132-149, 2008.

Stéphane Jeannenot, Pascal RICHARD, Frédéric RIDOUARD, Ordonnancement temps réel avec profils variables de consommation d'énergie, proc. Real-Time Embedded Systems, 2004.

Imran Rafiq Quadri, Abdoulaye Gamatié, Pierre Boulet, Samy Meftali, Jean-Luc Dekeyser: Expressing embedded systems configurations at high abstraction levels with UML MARTE profile: Advantages, limitations and alternatives. Journal of Systems Architecture - Embedded Systems Design 58(5): 178-194 (2012).

Frank Singhoff, Jérôme Legrand, Laurent Nana, Lionel Marcé: Cheddar: a flexible real time scheduling framework. SIGAda 2004: 1-8.