A Genetic Based Scheduling Approach of Real-Time Reconfigurable Embedded Systems

Hamza Gharsellaoui
National Institute of Applied Sciences and Technology (INSAT), Carthage University, Tunis, Tunisia
Higher School of Technology and Computer Science (ESTI), Carthage University, Tunis, Tunisia
Al-Jouf College of Technology, Technical and Vocational Training Corporation, Al-Jouf, KSA
gharsellaoui.hamza@gmail.com

Hamadi Hasni
National School of Computer Science (ENSI)
Manouba University, Tunis, Tunisia
hamadi.hasni@ensi.rnu.tn

Samir Ben Ahmed
Faculty of Science, Mathematics, Physics and Natural of Tunis (FST), Tunis El Manar University, Tunisia
National Institute of Applied Sciences and Technology (INSAT), Carthage University, Tunis, Tunisia
samir.benahmed@fst.rnu.tn

ABSTRACT

This paper deals with the problem of scheduling the mixed workload of both homogeneous multiprocessor on-line and off-line periodic tasks in a critical reconfigurable real-time environment by a genetic algorithm. Two forms of automatic reconfigurations which are assumed to be applied at run-time: Addition-Removal of tasks or just modifications of their temporal parameters: worst case execution time (WCET) and/or deadlines. 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 define an Intelligent Agent that automatically checks the system's feasibility after any reconfiguration scenario to verify if all tasks meet the required deadlines after a reconfiguration scenario was applied on a multiprocessor embedded real-time system. Indeed, if the system is unfeasible, then the proposed genetic algorithm dynamically provides a solution that meets real-time constraints. This genetic algorithm based on a highly efficient decoding procedure, strongly improves the quality of real-time scheduling in a critical environment. The effectiveness and the performance of the designed approach is evaluated through simulation studies illustrated by testing Hopper's benchmark results.

1. INTRODUCTION

The packing problems have been widely studied during the last three decades, as they are often faced in industry. The rectangular pieces packing problem, cutting also from rectangular board, is one particular case of this set of problems. The aim is often to achieve the minimum trim loss [15]. We had done some studies on packing problems in [10] and some other problems were studied in [16]. In this paper we propose a real-time genetic scheduling approach for the construction of optimization algorithms to the placement and scheduling problems such as packing and placement problems whose involve constructing an arrangement of items that minimizes the total space required by the arrangement. This is mainly due to the constraints imposed by the industrial applications, e.g. textile, wood, steel and embedded control systems. A recent survey on packing problems is given in [9]. In this paper, we specifically consider the two-dimensional (2D) rectangular strip packing problem based on a new real-time genetic scheduling approach, named RTGS algorithm. The input is a list of n rectangles that represents a set of n tasks with their dimensions (length and width that represent the worst case execution time and the deadline). The goal is to place the rectangles without overlap into a single reconfigurable device (shape) of width W and minimum height H. We further restrict ourselves to the oriented, orthogonal variation, where rectangles (tasks) must be placed parallel to the horizontal and vertical axes, and the rectangles can be rotated on each reconfiguration scenario. Further, for our test cases, all dimensions are integers. Like most placement problems, 2D rectangular placement (even with these restrictions) is NP-hard. Finally, our algorithm naturally solves a more general problem: given a set of tasks and a target rectangle, find a placement of a subset of those tasks which gives an optimal packing of the shape. This procedure is the our main contribution which is an efficient heuristic for online scheduling of hard real-time tasks to partially reconfigurable devices applicable to 2D area model. Numerical examples also showed the superiority of the proposed heuristic compared with the hopper benchmarks results. This paper is organized as follows. In Section 2, we provide the model assumptions and the related work. In Section 3, we present the resolution methods which use the bottom left algorithm and the guillotine constraint.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2598394.2605440 .

In Section 4, we show how our real-time genetic scheduling algorithm can be adapted for solving the general 2DPS problem. In Section 5, we undertake a comparative study of our proposed algorithm and evaluate its performance for the 2DPS problem using benchmark problems from the literature. Finally, in Section 6, we summarize the contributions of this paper and explain its possible extensions.

2. MODEL ASSUMPTIONS AND RELATED WORK

This section first presents the modeling of the reconfigurable resource (device) and the hardware tasks. Then, we introduce some related work in the 2D area model and we discuss practical limitations of this model under current technology. Finally, we introduce the contribution of this paper.

2.1 Definitions and Known Preliminaries

In this subsection, we will present the task model and the genetic algorithm definitions.

2.1.1 Task Model

A hardware task is a synthesized digital circuit that has been preplaced and prerouted. The task is stored in a position-independent way and can be relocated to different locations on the reconfigurable device by the operating system [6].
In our work, we will adopt this assumption and we will check for a good way to place and schedule hardware tasks based on their structural and timing characteristics.
According to [6], the structural characteristics of Operating Systems (OS) tasks are the size (area) and the shape. By this way, the authors consider that hardware tasks have a certain area requirements, given in numbers of reconfigurable units (RCUs). Like the most research works, we will model also in our work, the shape of a hardware task by a rectangle including all RCUs as well as the routing resources used by the task. Indeed, rectangular shapes simplify the task placement. In contrast, they lead to unused area inside the rectangle. In our original work, our goal is to minimize this unused area.
To formalize our work, the timing characteristics that might be known in advance are the arrival time a_i, the worst case execution time (WCET) c_i, and the deadline di of each task T_i such that $a_i + c_i \leq d_i$ and $1 \leq i \leq n$, of the n real-time tasks to be placed and scheduled in the shape. We assume that all tasks arrive at time $a_i = 0$, and we call them synchronous tasks.
Also, the structure model for each task T_i is a rectangular area of reconfigurable units given by its width and height, w_i x h_i. the hardware task area of the reconfigurable device is modeled as a rectangular area shape W x H of reconfigurable units.

2.1.2 Genetic Algorithm (GA)

A genetic algorithm (GA) is a procedure used to find approximate solutions to search problems through application of the principles of evolutionary biology. Genetic algorithms use biologically inspired techniques such as genetic inheritance, natural selection, mutation, and sexual reproduction (recombination, or crossover). For this problem, members of a space of candidate solutions, called individuals, are represented using abstract representations called chromosomes. The GA consists of an iterative process that evolves a working set of individuals called a population toward an objective function, or fitness function. The evolutionary process of a GA is a highly simplified and stylized simulation of the biological version. It starts from a population of individuals randomly generated according to some probability distribution, usually uniform and updates this population in steps called generations. Each generation, multiple individuals are randomly selected from the current population based upon some application of fitness, bred using crossover, and modified through mutation to form a new population [2].

2.2 Related Work

In general, packing problems are important in manufacturing settings; for example, one might need n specific rectangular pieces of glass to put together a certain piece of furniture, and the goal is to cut those pieces from the minimum height fixed-width piece of glass. The more general version of the problem allows for irregular shapes, which is required for certain manufacturing problems such as clothing production.
Nowadays, the problem of real-time placement and scheduling in the 2D area model is widely studied and has often been used in recent research works because it's not enabled by currently Field Programmable Gate Arrays (FPGAs) technology in contrast to the 1D model. Indeed, FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. For more information about the 1D and the 2D models, readers can see [6]. Indeed, the real-time scheduling of OS tasks in embedded systems requires efficient algorithms. In [3], [4], Brebner proposed an operating system approach for partial reconfigurable hardware. Furthermore, the problem of placement in the 2D area model has been proposed by Barzagan et al. in [1]. In [14], the authors discuss an online scenario where a resource manager schedules arriving tasks to a farm of FPGAs. Since each task occupies exactly one FPGA, there is no partial reconfiguration and placement is not an issue [6].
Burns et al., [5] describe several operating system functions, including a 2D transform manager that performs translation and rotation operations on tasks to better fit them to the device [6]. In [8] and [7], the authors investigate compaction, i.e., rearrangement executing tasks; in the 2D area model. In our work, we will adopt the rotation operation to find a best fit of the tasks in the shape. This method leads to maximize the contiguous free area and as a consequence, the chance of successful future placements of new arrival tasks was increased and the internal fragmentation was decreased.
In the related work, several algorithms and approaches were conducted with variants of the first-fit, best-fit, worst-fit and bottom-left (BL) bin packing algorithms.
In our work and based on our previously published conference contribution in [10], we will adopt the BL algorithm for the placement of hardware tasks, represented by a number of rectangular sub-tasks, with transformations that consist of a series of rotation and flip operations in order to reduce internal fragmentation.

2.3 Problems and limitations

According to the previous related work, we can deduce that; the main abstract is that tasks are modeled as relocatable rectangles that can be placed anywhere on the device in the 2D area model [6]. This relocatibility raises a number of investigations concerning the device homogeneity such as dedicated memories, task communication, embedded multipliers, real-time constraints and the partial reconfiguration. Task communication and real-time constraints are unresolved issue in the 2D area model where some tasks communicate with each other and with I/O devices need online routing and online delay estimation of their external signals. Furthermore, no analysis of the problem of communication infrastructure is provided.

2.4 Contribution of this Paper

The goal of our research problem is to minimize the waste and the total used space in the area model used. The area model used in our approach is the 2D model shown in figure 1, where the reconfigurable device is represented by a rectangular area of RCUs. So, our interest is to find a best placement approach based on a subset of tasks which gives an optimal placement of tasks inside the shape. However, to the best of our knowledge, the only work with the online placement and scheduling of OS tasks with the communication mechanism is that we propose in this original work. Our placement method is based on the BL algorithm with guillotine method, that we will see in details in the next subsection, which leaves some space between tasks and performs online routing of communication channels as a communication network that is not affected by user reconfigurations and rearrangement of tasks based on the genetic algorithm method. For this reason, we can believe of our proposed work advantages over the previous works in terms of scheduling performance. A significant improvement of the proposed approach results, over those obtained by the Hopper results, has been achieved.

3. GUILLOTINABLE TASK PLACEMENT

In this section, we explain the techniques used to resolve placement and scheduling (PS) problems based on the Bottom-Left heuristic which is a foundation work by [11]. For this reason, we describe it in some details.

3.1 Bottom Left (BL) Algorithm

The Bottom-Left (BL) heuristic was introduced in [11]. The placement Bottom Left (BL) algorithm using the permutation is executed to place rectangular pieces into the main sheet (object) in the two dimensional cutting and placement problems. In our work, we will adopt also this algorithm for the placement of real-time tasks on the reconfigurable device. We can describe the BL algorithm process using the following definition: This Bottom-Left placement algorithm takes a reconfigurable device (shape) and an input sequence of tasks and their allowable rotations (rotation criteria). The algorithm progresses placement approach by placing the first task in the lower left corner of the shape in its most efficient orientation (the orientation that yields the smallest bounding rectangle device height within the set of rotation criteria). With subsequent tasks, a valid location for placement is found by testing for intersections and containment. If the task is not intersecting by (or containing) other already placed tasks, then the location of the task is valid and therefore can be assigned to the shape. When a task is in a position that intersects with already assigned tasks, we use the rotation technique. The process continues as before with overlap/intersection tests and resolution until the task does not intersect and can be placed. Placement procedure with BL algorithm is completed when all tasks have been assigned to the reconfigurable device (shape).

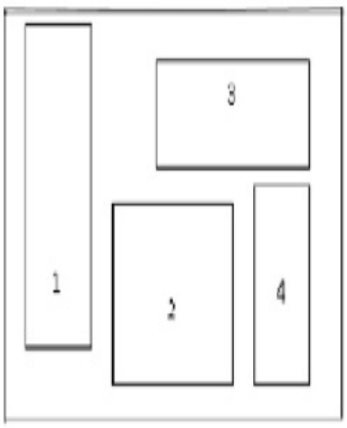

Figure 1: Guillotine Configuration.

3.2 Guillotine Placement Problems

The two-dimensional guillotine-cutting problem has been widely studied in the operational research literature. The unrestricted problem is known to be NP-hard. When cutting specific materials like glass it may be required that the rectangles can be cut out of the bin by a number of guillotine cuts which can be thought of as edge-to-edge cuts. The most common constraint requires guillotine cutting patterns; i.e., patterns where pieces can be obtained using a series of horizontal and vertical cuts. This constraint demands that all placed pieces are reproducible through a series of guillotine cuts that can cut the bin (pattern) into pieces so that each piece contains a box and no box has been intersected by a cut.

In our model placement and scheduling, the outer box represents the reconfigurable device which is represented by a rectangular area of reconfigurable units (RCUs) and the inner numbered boxes represent a hardware tasks. In our work also, we mean by a guillotine placement, the method through it when we apply a series of cuts through a reconfigurable device (shape) from one edge to the opposite edge and parallel to the other two edges of the shape in a straight line, there is no one task that can be deteriorated. That is, there should exist a series of face parallel straight cuts.

In our example, the first tasks placement is guillotinable and the solution to obtain a guillotine placement is feasible (Figure 1). In this figure, cut task 1, then task 3, then task 2 and the rest is the task 4 in order to obtain a feasible cut. This feasibility was assured in our work by the bottom left algorithm designed for a guillotine real-time reconfigurable scheduling specifically and only creates guillotine feasible cuts, while the second placement is not guillotinable and the solution to obtain a guillotine placement is unfeasible (Figure 2).

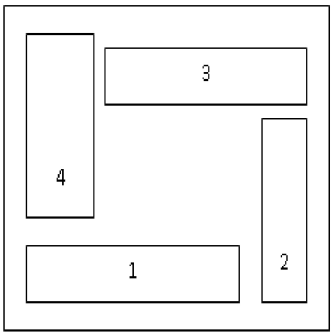

Figure 2: Non-Guillotine Configuration.

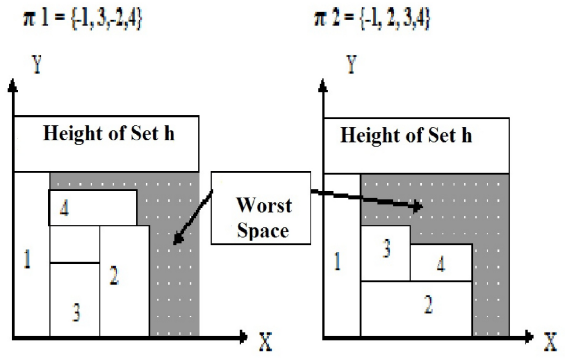

Figure 3: 2 possible permutations of tasks (1, 2, 3, 4).

Notation and example:
Alternatively, a task set to be placed in the shape can be represented by a permutation π [13]. $\pi = (i_1, ..., i_n)$ is a permutation and i is the index of the task (T_i). The permutation represents the sequence in which the tasks (rectangles) are placed. The advantage of this data structure is the easier creation of new permutations by changing the sequence. As a consequence of the variable data structure, every permutation has to be assigned to a unique placement reconfigurable device (shape).
Allocation of some of tasks with BL is illustrated in Figure 3.

In $\pi_1 = (-1, 3, -2, 4)$

task 1 is placed first to the bottom and then to the left as far as possible with a rotation of 90^o, (the sign - represents a rotation of the piece by 90^o). Tasks 3 and 2 are placed in the same manner with a rotation of the task 2. Then, task 4 is placed in its optimal location.

In $\pi_2 = (-1, 2, 3, 4)$

task 1 is placed first to the bottom and then to the left as far as possible with a rotation. Task 2 is then placed in its optimal location and tasks 3 and 4 are placed after that.

In addition, the cost of the BL-algorithm is $\Theta(n^2)$. This based on the fact, that each rectangle (task) T_i can be shifted a maximum of i times, because each shift is limited by one of the i - 1 placed rectangles (tasks) or by the corners of the shape. Hence, the cost of placing task T_i is Θ(i) and the whole cost amounts to $\Theta(n^2)$.

4. REAL-TIME GENETIC SCHEDULING APPROACH

A genetic search algorithm is a heuristic search process that resembles natural selection. There are many variations and refinements, but any genetic algorithm has the features of reproduction, crossover and mutation. Initially a population is selected, and by means of crossovers among members of the population or mutation of members, the better of the population will remain. In the case of our approach the real-time scheduling is based on genetic algorithm. For this reason we call this scheduling method as a real-time genetic scheduling approach.

4.1 Fitness-Function

For the genetic algorithm, the evaluation of a model set is obligatory, this is represented by a Fitness-Function f: $\pi \rightarrow R_+$ with the property $f(\pi_i) \geq f(\pi_j)$ if π_i is better than π_j.
The Fitness-Function value is inversely proportional to the height of a model set: $f(\pi) = 1/h(\pi)$;
Where $h(\pi)$ is the model set height follow to the permutation π created by the BL algorithm.
π represents one permutation (arrangement) of rectangular tasks. The following picture (figure 3) is an example of placement of 4 tasks (1, 2, 3, 4).

To reduce the internal waste we have developed two variants of the placement (arrangement) as shown in the figure 3 with the same height of the rectangle (shape) but different in quality. By logic they have not the same Fitness Function. With the last definition of the Fitness Function ($f(\pi) = 1/h(\pi)$) if two task set placement have the same height, their Fitness Functions are equal even if one is better than the other. For this reason, it's necessary to define another Fitness Function better than this one. Following the last picture (figure 3), it's clear that π_2 is better than π_1 because the remaining space in the result configuration is better with π_2. With this comparison, and if we consider only one objective which is the wastage minimization. Our Fitness Function is given by the following formula:

$F(\pi) = H(\pi) +$ Area (worst remaining space)$/h(\pi)$* width

Where:

- $H(\pi) = H - h(\pi)$, where H is a maximum given height that guarantee $H(\pi)$ be a positive value.
- $h(\pi)$ is the model set height follow to the permutation π.
- Area (worst remaining space) is the area of the remain-

ing space in the placement (arrangement) model following to the permutation π.
• width is a shape (reconfigurable device) width dimension.
• h(π)* width is the rectangle area occupied by the placed rectangles (tasks).

4.2 Initialization

It's evident that genetic algorithms uses m objects. Here the m objects are m permutations: π_1, π_2,.., π_m. We assign for each arrangement it's Fitness Function value: $f_i = f(\pi_i)$; i = 1, 2, ...,m.
We consider the permutation π_i and it's Fitness Function f_i together like one individual A_i: $A_i = (\pi_i, f_i)$.

4.3 The Tournament Selection

The principle of selection by tournament increases the chances for poor individuals (arrangements) to participate in the improvement of the population. The principle is very quick to implement. A tournament consists of a meeting between two arrangements (parents) at random in the population. The tournament winner is the arrangement better P_1 (parent P_1). We repeat this method to select the second best arrangement P_2 (parent P_2). The principle of the arrangement (parent) P selection is the following:
1. (P_1, P_2) = random($|n|$).
2. P = random(P_1, P_2).

4.4 Recombination

The crossover operator is special. Indeed, in order to respect the constraint that every task which is copied did not been repeated another time in order to respect the pieces unicity and creates two solutions or childs (permutations) $\pi_i new$ and $\pi_j new$ by combining two parents π_i and π_j. For each pair of parents (permutations), two crossover integers P_1 and P_2 are randomly chosen (generated) with the condition $1 \leq P_1, P_2 \leq n$. In the random position P_1, the crossover operator copy P_2 elements from π_i for the beginning of $\pi_i new$ and copy P_2 elements from π_j for the beginning of $\pi_j new$. Then, $\pi_i new$ is completed by the remaining elements of π_j with respect to the same order of appearance and $\pi_j new$ is completed also by the remaining elements of π_i with respect also to the same order of appearance.
Example
$\pi_i = (1, -2, 3, 4, -5, 6)$ and $\pi_j = (-6, 4, 2, 5, -3, 1)$

if ($P_1 = 2$ and $P_2 = 3$) then
$\pi_i new(1) = \pi_i(P_1) = \pi_i(2)$ = -2;
$\pi_i new(2) = \pi_i(P_{1+1}) = \pi_i(3) = 3$;
$\pi_i new(3) = \pi_i(P_{1+2}) = 4$;
$\pi_i new(4) = \pi_j(1)$ = -6;
$\pi_i new(5) = \pi_j(4) = 5$;
$\pi_i new(6) = \pi_j(6) = 1$;
Then, π_i new = (-2, 3, 4, -6, 5, 1). With the same method, π_j new = (4, 2, 5, 1, 3, 6). So, with the crossover processes we get m new permutations.

4.5 Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation is an important part of the genetic search when it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set to high, the search will turn into a primitive random search. For this case, we consider in our work the model of task set X composed by n permutations (arrangements) which are represented by a rectangular tasks. The mutation probability was fixed in our work to 0.05 as a good choice. We adopt the change of one task by another of task set as an elementary transformation and we evaluate the Fitness-Function f for each new placement set. This algorithm was proposed and evaluated in a series of numerical experiments that are run on problem instances taken from the literature, as well as on randomly generated instances which proven our approach.

5. EXPERIMENTATION RESULTS AND DISCUSSION

In this section we will present the experimentation results and the discussion part in order to demonstrate the performance of our method.

5.1 Simulation Results on Hopper Benchmarks

Our proposed algorithm was tested against the Hopper results. The results indicate that performance of our real-time genetic scheduling (RTGS) algorithm is better than Hopper algorithm represented by the low, average and better results. We now present experimental results demonstrating the effectiveness of our method for finding perfect placement task set. We use the benchmarks developed by Hopper. Those benchmarks contain collections with size ranging from 17 to 199 rectangles (tasks), all the instances are tested into a single rectangle as a reconfigurable device (shape) of width (W = 300) and minimum height (H = 250).

Algorithm		**RTGS (%)**	**Hopper (%)**		
Data	N	✓	Low	Average	Better
t1c.xls	**17**	93.12	91.62	92.67	83.47
t2a.xls	**25**	94.98	93.73	93.95	95.73
t2b.xls	**25**	92.96	92.61	93.10	88.16
t2c.xls	**25**	92.87	92.17	92.17	83.68
t3a.xls	**29**	95.32	93.46	93.90	90.91
t3b.xls	**29**	94.74	92.94	93.46	89.68
t3c.xls	**29**	93.42	93.15	93.46	88.46
t4a.xls	**49**	97.89	94.38	95.24	89.68
t4b.xls	**49**	96.78	95.10	95.24	89.68
t4c.xls	**49**	96.15	96.15	96.15	88.49
t5a.xls	**73**	95.78	95.14	95.24	93.89
t5b.xls	**73**	96.82	95.92	96.15	89.28
t5c.xls	**73**	95.33	95.14	95.24	88.10
t6a.xls	**97**	96.48	96.75	96.75	94.48
t6b.xls	**97**	96.79	95.64	96.15	93.89
t6c.xls	**97**	96.34	95.92	96.15	95.69
t7a.xls	**199**	97.56	97.09	97.09	95.23
t7b.xls	**199**	97.69	96.36	96.93	96.15
t7c.xls	**199**	98.71	96.69	97.14	93.90

Table 1: Comparison results applied to Hopper Benchmarks

Results are conducted by many runs (20 runs) and we are taken the average performance results to present the effectiveness of our algorithm RTGS against the hopper results [12].

5.2 Discussion

We evaluate our hybrid algorithm on the guillotinable instances from this set by an interesting comparison between the running results which are summarized in Table 1. This table reports, for each collection of instances (files) the number of items or placed rectangles (a set of tasks in our case study), and the average of occupied space of Hopper results (Hopper (%)) and our real-time genetic scheduling algorithm (RTGS (%)). By Examining Table 1 and considering only those instances, we prove the performance of our proposed algorithm and we observe that the real-time genetic scheduling algorithm (RTGS (%)) performs better than Hopper results (Hopper (%)), and offer more gains in terms of occupied space, and minimum waste of shape.

6. CONCLUSION AND FUTURE WORK

In this paper we have proposed a real-time genetic scheduling approach for solving the constrained two-dimensional placement (2DP) problem. Starting from a packing with height H, this approach tries to solve the (2DP) problem with decreasing values of H while avoiding solutions with tall and thin wasted spaces. A specific fitness function f 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. Since we were unable to find a counter example for which the approach fails, we conjecture that it always finds an optimal constrained guillotine cutting. This should be extended to include other types such as asynchronous arrival time and aperiodic tasks problems. Further research in this area will include studying the impact of rectangular devices representing tasks on guillotine placement and scheduling problems.

7. ACKNOWLEDGMENTS

The authors would like to thank the reviewers of this work for their valuable comments.

8. REFERENCES

[1] K. R. Bazargan, K. and M. Sarrafzadeh. Fast template placement for reconfigurable computing systems. *IEEE Design and Test of Computers*, 17(1):68–83, May 2000.

[2] J. E. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *Operational Research Society*, 36:297–306, 1985.

[3] G. Brebner. A virtual hardware operating system for the xilinx xc6200. In *Int'l Workshop Field-Programmable Logic and Applications (FPL) Proceedings*, pages 327–336, 1996.

[4] G. Brebner. *The Swappable Logic Unit: A Paradigm for Virtual Hardwarel*. 1997.

[5] D. A. H. J. S. S. Burns, J. and M. Wit. A dynamic reconfiguration run-time system. pages 66–75, 1997.

[6] H. W. Christoph, S. and M. Platzner. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. *IEEE Transactions On Computers*, 53(11):1393–1407, 2004.

[7] E. H. M. M. S. H. Diessel, O. and B. Schmidt. Dynamic scheduling of tasks on partially reconfigurable fpgas. *IEEE Proc. Computers and Digital Techniques*, 147(3):181–188, May 2000.

[8] M. Diessel and H. ElGindy. *On Scheduling Dynamic FPGA Reconfigurations*. 1998.

[9] K. Dowsland and W. Dowsland. Packing problems. *European Journal of Operational Research*, 56:2–14, 1992.

[10] H. Gharsellaoui and H. Hasni. On a genetic-tabu search based algorithm for two-dimensional guillotine cutting problems. *IJAPUC*, 4(2):26–40, 2012.

[11] P. Gilmore and R. Gomory. The theory and computation of knapsack functions. *Operations Research*, 147:1045–1074, May 1996.

[12] E. Hopper and B. Turton. Problem generators for rectangular packing problems. *Studia Informatica Universalis*, 2(1):123–136, 2002.

[13] S. Jakobs. On genetic algorithms for packing polygons. *Euro. J. of Op. Res.*, 88:165–181, 1996.

[14] T. K. Y. V. S. J. Jean, J.S. and R. Cook. Dynamic reconfiguration to support concurrent applications. *IEEE Trans. Computers*, 48(6):591–602, June 1999.

[15] H. Teng and D. Liu. An improved bl-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112:413–420, 1999.

[16] Y. Z. Teng, H.F. and X. Gao. *Layout optimization for the dishes installed on a rotating table*. Science in China (series A), 1994.