# SCHEDULING APPLICATION USING PETRI NETS : A CASE STUDY: INTERGRÁFICAS S.A.

**Gonzalo Mejía Delgadillo, Ph.D., Sebastián Poensgen Llano, M.Sc.**
Department of Industrial Engineering, Universidad de los Andes, Bogotá, Colombia

**Abstract**
In most manufacturing settings scheduling is a difficult task due to the complexity of the system. Hence powerful tools that can handle both modeling and optimization are required. Most of the research in this area focuses in either developing optimization algorithms, or in modeling complex production systems. However, few tools are aimed to the integration of both of them. In this paper, a Petri Net-based integrated approach, for simultaneously modeling and scheduling manufacturing systems, is proposed. A prototype that simulates the execution of the production plan, and implements priority dispatching rules to solve the eventual conflicts, is presented. Such an application was tested in a printing company in Colombia (South America) that has complex flexible job shop-type system. Experiments on the real system were conducted, and results show interesting benefits when comparing with the current scheduling policies.

**Keywords**:
Scheduling, Petri Nets, Dispatching Rules. Printing

## 1 INTRODUCTION

Performance in most manufacturing settings is affected by operative decisions related to job scheduling such as: a) selection of jobs in queue, b) priority when choosing a machine for a job processing (among parallel machines), and c) assignment of resources in the execution of a production plan. Such decisions have significant impact on systems efficiency, operational costs and service promise fulfillment, and are frequently taken intuitively by the operators, based on their experience.

In order to support this decision process and guarantee the system efficiency, computational applications need to be developed. Such applications should be able to (i) provide modeling aids, (ii) apply scheduling techniques and (iii) define a production plan. This paper presents a Petri Net-based prototype application which is tested on a Colombian printing company.

Petri Nets are well known for their modeling potential, and for their ability to implement optimization techniques. Karl Petri developed this technique in 1962 for communication system analysis. Its use has been extended to other application fields, like manufacturing [1].

In the industry, Petri Nets were mainly used for modeling manufacturing plants as discrete event systems. In the seventies the GRAFCET tool (Petri Net-based) was introduced in order to specify, validate and implement logic controllers in production systems. GRAFCET has been recognized all over Europe and implemented in many countries [2]. Petri Nets have also been used in system design, and modeling, at companies like Microsoft Corporation, AT&T, Digital Equipment Corporation and Applied Materials Inc. [3].

When Petri Nets were introduced, many papers were published: Topics included resource utilization, bottlenecks, throughput, cycle times and capacity estimations [4] [5]. In order to validate the use of Petri Nets, manufacturing systems were evaluated using different techniques like simulation, queuing theory, probability and stochastic Petri Nets [6].

On the other hand, the Petri Net potential for analyzing and modeling complex systems has encouraged its use on scheduling problems. A Beam Search algorithm was implemented on Petri Nets in order to find an optimal production schedule [7]. Later, the Branch and Bound method was used in robot task programming, truncating the net in smaller sub-nets. This technique was complemented with dispatching rules for selecting the firing transition [8] [9]. A heuristic that only generates part of the Petri Net reachability graph (the graph deploying all the net possible states) was suggested, presenting three searching types: depth search, bread search and a mixture of both [10]. In 1994, an A* Search type heuristic was tested in the selection of a schedule for a flexible job shop. This approach reduced the computational effort, but sacrificed the solution quality [11]. Further on, an algorithm that combines the A*Search with a node selection strategy was suggested for generation scheduling programs, conducting to near optimal results with moderated computational efforts [12].

Literature surveys show that research in this area focuses either in developing algorithms for solving scheduling problems, or in designing and using tools to model complex production systems. Few projects are aimed for the integration of both. However, an approach of a scheduling software using Petri Nets was recently presented, in which the modeling platform is separated from the scheduler. In this case an interface between the engine and the Petri Net-based sequencer is suggested [13].

This document describes how the modeling approach proposed in [12] was translated into a real-life printing plant. First production plans are modeled using Petri Nets. Then a number of dispatching rules were implemented for solving eventual scheduling conflicts (selection of jobs in queue, machine priorities). When running the application a detailed resource schedule is obtained, describing both the resource occupation during simulation time (Gantt Chart), and several general system performance measures. The user should compare results using different dispatching rules, and determine which schedule suits her better.

This scheduler’s prototype was tested in a printing company in Colombia which has a complex flexible job shop-like configuration. Most workstations have parallel machines, and as jobs are exclusive for each customer, many different processing routes are handled. Experiments on different production plans were conducted, comparing the actual scheduling policy with the results of the application, finding interesting benefits for the company.

## 2 SIMULATION WITH PETRI NETS

### 2.1 Introducing Petri Nets

Most of the development of Petri Nets emerged by the need of specifying synchronization situations, asymmetric systems (alternative routes) and shared resource conflicts; essential events in the representation of a manufacturing setting.

A Petri Net is a set of nodes and arcs. There are two types of nodes: places and transitions, which represent the state of the system and the occurrence of events, respectively. In manufacturing systems, places would represent operations (e.g., process, transportation, reparation), and transitions symbolize events, such as termination of a job processing or a machine breakdown. Arcs are directed, and connect places with transitions (or transitions with places). Tokens reside in places and represent the truth of the condition associated with such a place. The firing process induces a token's flow among places; when a transition fires, tokens from all its input places are removed and put into the transition output places. A transition can only be fired if it has been enabled (i.e. there are sufficient tokens at its input places).

There are many different types of Petri Nets, though in this project only Marked Timed-Place Petri Nets (TPPNs) are of concern. Timed-Places are useful for modeling processing times, flow times or breakdowns, meaning that an amount of time may elapse until tokens enable their output transitions.

A formal definition of Timed Petri Nets [12] is:

$G = (P, T, I, O, M_o, M_{r0,} \tau)$, where:

- P is a set of places graphically represented by circles.
- T is a set of transitions graphically represented by bars, with $P \cup T \neq 0$ and $P \cap T = 0$.
- I is a function that specifies arcs going from transition to places $P \times T \rightarrow \{0,1\}$.
- O is a function that specifies arcs going from places to transitions $P \times T \rightarrow \{0,1\}$.
- $M_0$ is the initial marking where values at each position represent the number of tokens at the *i*th place (graphically represented by dots).
- $M_{r0}$ is the initial remaining time vector, which contains the time left for tokens to enable the corresponding output transition.
- $\tau$ is the set of time delays associated with places.

Token flow among places describes the dynamic behavior of the net, conducting to its analysis and study. This behavior is regulated by transition enabling and firing rules. For this type of Petri Net, a transition $t \in T$ is enabled, if every input place from has at least 1 token. Every enabled transition may be fired. When a transition fires, the net changes its state inducing a new marking M'. In such cases it is said that M' is reachable from M [14].

An example (Example 1), of two jobs being processed by the same machine, is presented, to show how Petri Nets work (Figure 1). When tokens are in $P_0$ and $P_3$, job 1 and job 2 are ready to be processed. $P_1$ symbolizes the processing of job 1 and $P_4$ the processing of job 2. Note that these processes share the same resource (machine 1) identified as $P_6$. The number of tokens at $P_6$ represents the resource capacity. If no tokens are in $P_6$, it means that the resource has no remaining capacity and arriving jobs must wait in queue. If this is the case, the resource is being used on some job and either $P_1$ or $P_4$ would have a token, meaning that the corresponding job is being processed. Tokens in places $P_2$ and $P_5$ represent the termination of job 1 and job 2, respectively.

Changes in the system depend on the sequence of fired transitions. At the initial state, transitions $T_0$ and $T_2$ are enabled. When executing the net, one of them must be selected and fired. If $T_1$ is selected the machine will process job 1 first, otherwise job 2 goes first. Firing one of these transitions will change the state of the system, and a new set of transitions will be enabled. Illustration of each step of the execution of this example is shown below.

At the initial state (Figure 1a.), tokens at $P_0$ and $P_3$ means that both jobs are ready to be processed by machine 1. Transitions $T_0$ and $T_2$ are enabled. $T_0$ is arbitrarily selected.

When $T_0$ is fired (Figure 1b.), the system state changes, meaning that job 1 is being processed. $P_1$ has an associate processing time. Once this time is exhausted $T_1$ is enabled. As $T_2$ is no longer enabled, $T_1$ is the only next enabled transition.

Firing $T_1$ (Figure 1c.). This new state enables $T_2$ and means that job 1 is finished. $T_2$ will be fired.

Firing $T_2$ (Figure 1d.). Machine 1 processes job 2 during $P_4$ associated time. Transition $T_3$ is enabled and is the next and last transition to be fired.

Firing $T_3$ (Figure 1e.). Job 2 is finished and machine 1 is idle again. No more transitions are enabled, and the simulation is over.

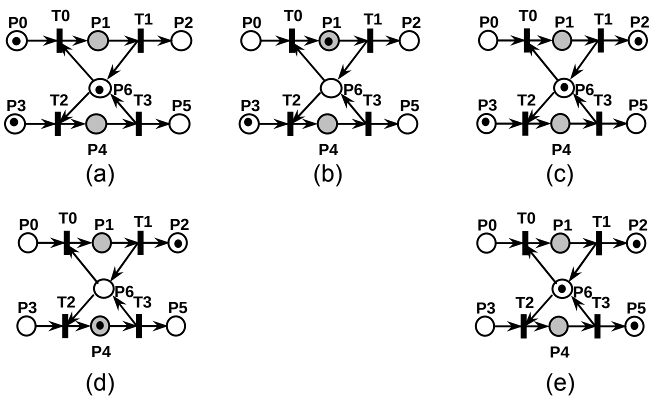

Figure 1: Machine processing two jobs.

If transition $T_2$ is fired first, the processing sequence would change. This fact could for example impact the due date fulfillment.

## 2.2 Executing (Simulating) the net

The execution algorithm corresponds to the firing of transitions until the final state is reached.

The execution algorithm follows the next steps:

1. Define the initial state marking $M_0$.
2. Identify the enabled transitions (u).
3. Select a transition, say $u_j$, from vector (u), using some selection criteria.
4. Fire the transition.
   4.4. Remove tokens from input places of transition $u_j$.
   4.5. Calculate the time elapse for firing transition $u_j$.
   4.6. Put tokens into the output places of transition $u_j$.
5 Update the marking $M_i$ and remaining process time $M_r$ vectors.
6 Advance the system clock.
7 If $M_i$ is equal to $M_f$ (final state) then stop, otherwise go to step 2.

As each fired transition defines a new array of enabled transitions (vector u), the firing sequence for reaching $M_f$ may change considerably. Scheduling concepts may be involved in the selection criteria (step 3) in order to decide which transition fires.

The existence of several enabled transitions could represent operational conflicts. A typical conflict occurs when two or more jobs are competing to use the same resource (as in Example 1) or when a single job must select between two or more machines (flexible systems). In this approach dispatching rules are proposed for solving these conflicts. The dispatching rules implemented in this model are well known in the scheduling literature [15] and are listed bellow:

1. SPT (shortest processing time).
2. LPT (longest processing time).
3. WSPT (weighted shortest processing time).
4. EDD (earliest due date).
5. CR (critical ratio).
6. MS (minimum slack).
7. ATC (apparent tardiness cost). The expression I(t) must be calculated for each job in queue.

$$I_i(t) = (w_i / d_i)\exp(-\max(d_i - p_i - t;0) / K * \bar{p} \qquad (1)$$

Where K is an empiric factor, which is assumed to be 5, value suggested by some authors. $\bar{p}$ is the average of the processing times $p_i$ of every job in the queue. The job in the queue with the greatest $I_i(t)$ has priority [16].

## 3 SCHEDULING APPROACH

### 3.1 Program structure

The scheduler prototype was developed in JAVA®. The program reads the information of the Petri Net-model of the respective production plan and creates a Gantt graph for visualizing the job sequences in the respective machines. The dispatching rule must be specified by the user.

The program executes the following steps:

1. Creates an instance of a Petri Net object
2. Creates an instance of the object Scheduler and selects the firing transition rule.
3. Performs the Petri Net execution algorithm until it reaches the final state.
4. Generates the system indicators, the Gantt graph and prints the output file.

The system performance measures are:

- Late jobs: Register the jobs that finish late.
- Total Weighted tardiness: Sum of tardiness weighted by jobs priorities.
- Makespan: Maximum difference between the start and the finish time of every job.
- System total time: Elapsed time between the start of the first job and the finish of the last.
- Resource utilization: % utilization time of each machine during the total system time.

The first two indicators measure the service quality and the last three describe the system's efficiency.

### 3.2 Example

Model construction and application's prototype are described in Example 2 (Figure 2.), where two jobs with alternative routes (flexible system), synchronization and shared resource utilization are processed,

For selecting jobs in queues, it is necessary to specify job due dates and priorities, and for the machine priority problem, information related to machines is needed, such as processing quality an~~d~~ functional reliability. See table 1. In this example, machine quality and reliability are equal for all machines.

Table 1: Due dates and priorities.

| Job | Due date | Priority |
|---|---|---|
| job 1 | 13 time units | 1 |
| job 2 | 15 time units | 2 |

The processing of the two jobs can be described as follows: Job 1 is printed (10 min), cut (8 min) and packed (3 min). Job 2 has two parts, sheets and cover. Sheets are printed (5 min), cover is folded (3 min) and after assembling, the product is packed (4 min). A detailed description of each place of the net is shown in table 2. At initial state tokens are placed at both the start places and resource places, meaning that the system is ready to begin processing. When tokens reach the termination places $P_7$ and $P_{15}$ after a sequence of fired transitions, the simulation is finished. In this example, conflicts appear at the printing process (jobs in queue) and cutting process (alternative routes).

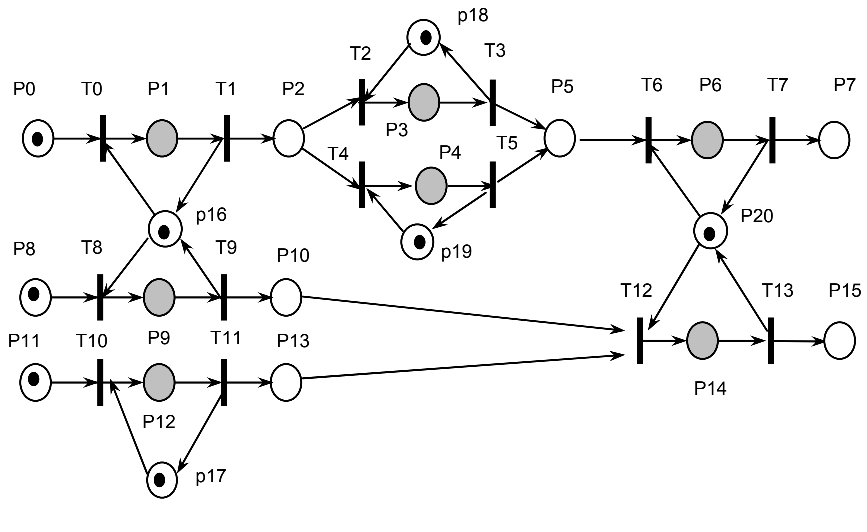

Figure 2: Production system with alternative routes, synchronization and shared resources.

Table 2: Places description for Example 2.

| | | | | | |
|---|---|---|---|---|---|
| P0 | Job 1 in queue at printing WS | P7 | Job 1 is finished | P14 | Packaging WS processing job 2 |
| P1 | Printing WS processing job 1 | P8 | Job 2-Sheets in queue at printing WS | P15 | Job 2 is finished |
| P2 | Job 1 in queue at cutting WS | P9 | Printing WS processing Job 2-Sheets | P16 | Printing WS |
| P3 | Cutter A processing job 1 | P10 | Job 2-Sheets in queue at packaging WS | P17 | Folding WS |
| P4 | Cutter B processing job 1 | P11 | Job 2-Cover in queue at folding WS | P18 | Cutter A |
| P5 | Job 1 in queue at packaging WS | P12 | Folding WS processing Job 2-Cover | P19 | Cutter B |
| P6 | Packaging WS processing Job 1 | P13 | Job 2-Cover in queue at packaging WS | P20 | Packaging WS |

The user selects a dispatching rule, and after running the program, a Gantt graph and the system measures are generated. Running this example for dispatching rules EDD and SPT provide the results presented in Table 3.

The SPT rule requires a longer total time compared to the EDD rule but the SPT rule shows just 1 late job while the EDD rule shows 2 late jobs. On the other hand, EDD only uses Cutter B and SPT only uses Cutter A. These results show that the use of different dispatching rules has a significant impact on the system performance.

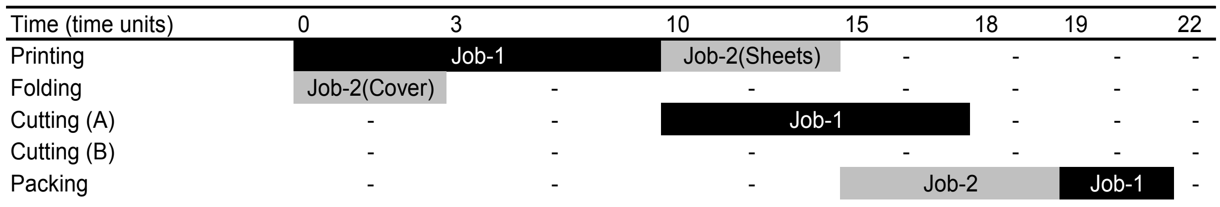

Figure 3: Results for Example 2 using EDD.

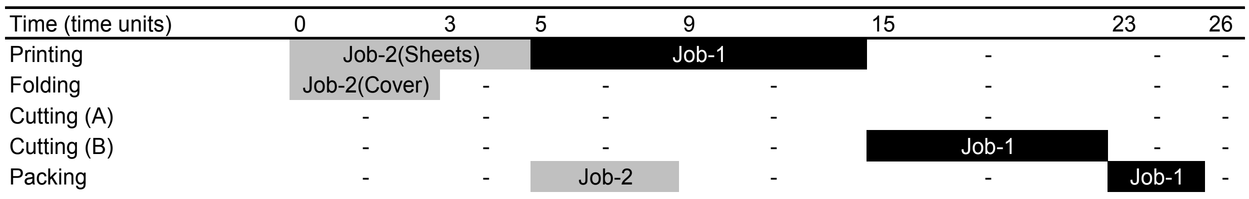

Figure 4: Results for Example 2 using SPT.

Table 3: Compared results for example 2.

| Performance measures | EDD | SPT | Resource utilization | EDD | SPT |
|---|---|---|---|---|---|
| Late jobs | 2 | 1 | Printing WS | 68.2% | 57.7% |
| Total weighted tardiness | 13 | 11 | Folding WS | 13.6% | 11.5% |
| Makespan (time units) | 22 | 21 | Cutter A | 36.4% | 0.0% |
| System total time (time units) | 22 | 26 | Cutter B | 0.0% | 30.8% |
| | | | Packaging WS | 31.8% | 26.9% |

## 4 CASE STUDY

### 4.1 Model construction

In order to test this prototype application, several experiments were conducted at Intergráficas S.A., a small printing company in Bogotá, Colombia. The manufacturing setting consists of 17 workstations, some of them with parallel machines. Manufactured items follow a make to order scheme, and many different processing sequences are handled. Such a system implies many operative decisions making job status visibility and finish date estimation a difficult task. For the company, maintaining service levels is a priority factor.

Jobs go through the system, following pre-defined processing sequences. Processing times depend normally on both the complexity and the size of the job order. For example, a two-side lamination of 1000 greeting cards takes longer than the lamination of 1.000 posters.

Experiments were conducted over three real production plans. Each plan fed the system for 3 days and includes ready to process and already-in-process jobs. PP-1 (production plan 1) has 18 jobs, PP-2 has 14 and PP-3 has 9 jobs. Different production plan sizes were chosen in order to analyze if this factor has an effect on the results.

For each PP two models were built: one for representing the actual policy (with some routes restrictions), and other for using the Petri net approach. The current scheduling policy includes the following criteria:

- Jobs ready to be processed are ordered according to their due dates.
- Top-client jobs have been identified and have priority for processing.
- A printer is pre-assigned for each job according to the job requirements (printing quality, speed) and availability of printers. In addition the shop scheduler attempts to balance the machine utilization at the printer workstation. This assignment is also done for other workstations like stamping or cutting.

The assignment process is done intuitively, and represents alternative route restrictions. Other decisions taken by the operators include the following criteria:

- Top-clients have priority for job selection and in general, the job with nearest due date is preferred. Ties are broken by selecting the job with the shortest processing time.
- Idle machines are preferred for selection. If many machines are idle, the one with higher processing reliability has priority. Ties are broken by choosing the machine with better processing quality. If the conflict is not solved, fastest machines (depending on machine speeds) are selected.

For this approach, model construction follow $S^3PR$ (simple sequential process resources) rules, structure that guarantees no net-blocking [12] [17]. Buffers are used to prevent the occurrence of deadlocks. Under this structure there is no net-bounding when executing the simulation [18] [19].

In addition, the following assumptions were considered:

1. Material and other processing resources (personal, tools) are available 100%.
2. Machine breakdowns, and unexpected events, are not being considered in the simulation.
3. Transfer times between workstations are negligible.
4. A resource can only process one job at a time, and processing interruptions are not allowed.

### 4.2 Experimental results

For each PP, 8 experiments were run: 1 for the actual policy and 7 using dispatching rules. Running any of these models in a computer with Intel® Pentium® 4, 512 MB of RAM and Microsoft Windows XP, takes less than 4 seconds. Results are shown for the three PP's.

Table 4: Experimental results.

| Results | PP-1 | | | | PP-2 | | | | PP-3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rule | L.J. | TWT | Cmax | S.T. | L.J. | TWT | Cmax | S.T. | L.J. | TWT | Cmax | S.T. |
| Actual | 3 | 2704 | 6393 | 6475 | 4 | 4846 | 1606 | 1886 | 0 | 0 | 2738 | 2738 |
| WSPT | 3 | 2611 | 6393 | 6539 | **4** | **3822** | **1569** | 1849 | 0 | 0 | 2718 | 2718 |
| SPT | 3 | 2614 | 6393 | 6605 | 4 | 4626 | 1593 | 1758 | 0 | 0 | 2696 | 2718 |
| MS | **3** | **2440** | **6393** | 6539 | 4 | 3894 | 1642 | 1922 | 0 | 0 | 2738 | 2738 |
| LPT | 5 | 4210 | 6393 | 6393 | 4 | 5022 | 1642 | 1757 | **0** | **0** | **2623** | 2738 |
| EDD | 3 | 2599 | 6393 | 6539 | 4 | 3894 | 1642 | 1922 | 0 | 0 | 2738 | 2738 |
| CR | 4 | 2680 | 6393 | 6393 | 4 | 3984 | 1642 | 1922 | 0 | 0 | 2692 | 2738 |
| ATC | 3 | 2623 | 6393 | 6539 | 4 | 4068 | 1642 | 1922 | 0 | 0 | 2738 | 2738 |

The use of an arbitrary dispatching rule may achieve good results for some measures, but poor results for others. Schedule selection depends on the priorities of the company. As for Intergraficas S.A., on-time delivery is more important than efficiency. Thus this measure has a higher priority.

For PP-1, the recommended schedule was obtained with MS, which has best performance in late jobs, total weighted tardiness and cycle time. Using the schedule obtained with LPT would take 146 minutes less (reduction of 2.23%), but as efficiency is not as important as service, the MS rule suits better.

For PP-2 the schedule obtained using WSPT seems to be the best option. This rule produces better comparative results in terms of late jobs, total weighted tardiness and cycle time. LPT would reduce total time in 8.92% (165 minutes) comparing to WSPT, but using this rule would produce a bad performance in the rest of measures.

PP-3 is the smallest plan, having fewer conflicts than the others. This fact implies that similar results are obtained for different rules. However, the schedule followed with LPT is recommended. It presents better results in terms of service indicators and maximum cycle time. WSPT and SPT would give a shorter total system time (20 minutes reduction -> 1%), a fact that could be ignored.

Resource utilization, at workstations with parallel machines, could be an important factor when selecting the best schedule. At the printing process for example, some schedules use the least reliable printer intensively, a fact that could be considered for selecting other schedule. In PP-1, WSPT, SPT and LPT tend to use tweezers-1 (best quality), while other rules use tweezers-2. The utilization of machines within workstations may change, but the busiest workstation of the system will not vary for the different schedules. Although the busiest workstation changes depending on the production plan; for PP-1 the tweezers is the busiest workstation, for PP-2 is lacquering and for PP-3 is lamination.

The use of different dispatching rules conducts to schedules with significant differences in the system measures. No pattern was identified for concluding that the use of a particular rule is recommended. It will depend on the company's preferences. In general, the dispatching rules that showed better results were MS, WSPT, EDD and LPT. The user should run each dispatching rule, analyze results and select the better schedule. The production plan size has an important impact in the application's utility.

## 5 CONCLUSIONS

This paper has presented an application of Petri Nets to a real life manufacturing setting. The modeling power of Petri Nets in combination with dispatching rules might bring significant benefits for this company. For instance:

1. Evaluate and compare different schedules for the same production plan, according to the company's preferences.
2. Visualize job schedules and estimate job termination times to support service commitments, and offer better service quality to customers.
3. Visualize the workstation/machine utilization. Identify and optimize bottlenecks with an efficient resource assignment. Utilizations might be an important decision's criteria for selecting the best schedule.
4. Measure and monitor performance indicators for implementing continuous improvement.
5. Test different options, changing parameters such as processing times, production plan sizes, alternative routes or disable/enable resources for supporting investments, and operative decisions.

Particularly, for Intergráficas S.A. the implementation of this computer application (based on the conducted experiments) would represent the following benefits:

1. Production time reduction of more than 5%. The plant works 70 shifts per month (18 in extra-time). This reduction represents at least 3 extra shifts less per month (U$D 4.500 / month).
2. Better service quality. The estimation of job termination times allow to inform customers about due dates, reducing tardiness cost.
3. The programming of maintenance activities during operational time (saving additional maintenance shifts).
4. The scheduling process doesn't depend on the intuition of the production assistant or the operators.
5. Planning decisions as machine investments could be supported by testing the system at different conditions.

The implementation process of such this application at Intergráficas S.A. requires the following steps: a) integration with the current information system, b) redefinition and redesign of the scheduling process including the use of the application and c) personal training for its use.

Further research will be focused on using more complex optimization scheduling algorithms and on improving user's interface. The research agenda includes:

- Develop the application in terms of computational efficiency and user's interface. Modeling using xml could be an interesting approach.
- Implementation of optimization algorithms such as Beam Search, A*Search [12], and Branch and Bound algorithms is suggested.
- Test the application at other manufacturing settings as flow shops, job shops or flexible manufacturing cells. Explore Petri Nets potential for modeling complex systems.
- Lead comparative research between this application and other scheduling tools.

## 6 REFERENCES

[1] Zhou M. C., DiCesare F., 1993, Petri Net synthesis for discrete event control of manufacturing systems, Kluwer Academic Publisher, (USA), 187 – 188.

[2] Guia A., DiCesare F., 1993, GRAFCET and Petri Nets in manufacturing systems, Intelligent Manufacturing: Programming Environments for CIM, Springer Verlag, London, 153 – 176.

[3] Zhou M. C., Venkatesh M., 1999, Modeling, simulation and control of flexible manufacturing systems, Intelligent Control and Intelligent Automation, Vol 6. World Scientific Publishing Co., Singapore, 43 – 44.

[4] Zhou M. C., DiCesare F., 1991, Parallel and sequential mutual exclusions for Petri Net modeling for manufacturing systems, IEEE Trans. on Robotics and Automation, Vol 7, No. 4, 515 – 527.

[5] Dubois D., Stecke K., 1983, Using Petri Nets to represent production processes, Proc of the 22nd IEEE Conf. on Decision and Control, San Antonio, TX, 1062 – 1067.

[6] Watson J.F., Desrochers A. A., 1991, Applying GSPN to manufacturing systems containing non-exponential transition functions, IEEE Trans. on System. Man. and Cyber. Vol 21, No. 5, 1008 – 1017.

[7] Shih J., Sekiguchi T., 1991, A timed Petri Net and beam search based on-line FMS scheduling system with routing flexibility, Proc. of the 1991 IEEE Int. Conf .on Robotics and Automation. Sacramento, CA, 2548 – 2553.

[8] Shen L., Chen Q., Luth J. Y., Zangh Z., 1992, Truncation of Petri Net models of scheduling problems for optimum solutions, Proc. of Japan/USA Symposium on Flexible Automation, 1681 – 1688.

[9] Zhou M. C., Chiu H., Xiong H. H., 1995, Petri Net scheduling of FMS using branch and bound method, Proc. of 1995 IEEE Int. Conf on Industrial Electronics, Control and Instrumentation. Orlando, FL, 211 – 216.

[10] Zhou M. C., Venkatesh M., 1999, Modeling, simulation and control of flexible manufacturing systems, Intelligent Control and Intelligent Automation, Vol 6. World Scientific Publishing Co., Singapore, 56.

[11] Sun T. H., Cheng C. W., Fu L. C., 1994, A Petri Net based approach to modeling and scheduling for an FMS and a case Study, IEEE Trans. on Industrial Electronics, Vol 41, No. 6, 593 – 601.

[12] Mejía G., Odrey N., 2005, An approach using Petri Nets and improved heuristic search for manufacturing system scheduling, Journal of Manufacturing Systems, Vol 24, No. 2, 103 – 117.

[13] Mejía G., Medaglia A., Gutierrez E., 2006, A framework for integrating shop floor modeling and scheduling applications: an illustrative case, Working paper, Department of Industrial Engineering, Uniandes.

[14] Zhou M. C., Venkatesh M., 1999, Modeling, simulation and control of flexible manufacturing systems, Intelligent Control and Intelligent Automation, Vol 6. World Scientific Publishing Co., Singapore, 70.

[15] Hahmias S., 1997. Production and Operations analysis. 3rd. Edition, Irwin/Mc. Graw-Hill, USA, 404.

[16] Pinedo M., 2002, Scheduling: Theory, Algorithms, and Systems, Prentice Hall, USA, 34 - 55.

[17] Ezpeleta J., Colom J.M., Martínez J., 1995, A Petri Net based deadlock prevention policy for flexible manufacturing systems, IEEE Transactions on Robotics and Automation. Vol. 11, No. 2.

[18] Mu J., 1997, PetriNets for modeling automated manufacturing systems with error recovery, IEEE Transactions on Robotics and Automation, Vol. 13, No. 5.

[19] Mejia G., 2002, An Intelligent Based Architecture for Flexible Manufacturing Systems Having Error Recovery Capacity, PhD Thesis. Department of Industrial Engineering, Leigh University.