Content-Driven Neural Network Design of a PSP

I-Jong Lin, Eric Hoarau and Jun Zeng; Hewlett Packard Laboratories; Palo Alto, CA, USA

Abstract

The PSP design technique that we present is, at its core, the generalization of structured job data via neural network learning techniques. While current PSP design focuses on the optimal use of capital equipment as its primary motivation, the essential competitive advantage of digital presses and workflow is its ability to adapt to different types of content with highest robustness to failure and minimal component-level change; these characteristics are also the same for neural networks. By generalizing the fulfillment order with a representative neural network, we can automatically identify redundancy between jobs and optimize the infrastructure for a particular content mix. By adaptively changing the neural network in the face of different job fulfillment demands, the neural network can also indicate how to transform the current PSP infrastructure to handle a new mix of jobs requests. We apply a structural learning technique based on a subset of Hidden Markov Models, Directed Acyclic Graphics, and then map these neural structures into print shop infrastructure. We will demonstrate our results with real world PSP data, and compare and contrast the current real world PSP design with its neurally designed counterpart.

Introduction

The point of the paper is to draw comparisons between neural network analysis and digital PSPs. We define the term a *digital PSP* from a broad range of PSP; a digital PSP closely integrates IT infrastructure, digital presses and physical processing devices such as binders and cutter. PSPs are given a set of orders and must process these orders within certain time and cost constraints. In particular, a digital PSP clearly has an IT infrastructure in place to primarily source job orders from the Internet, but also has significant capital investment in physical infrastructure (presses, finishers, binders, etc.).

Since a digital PSP sources the majority of its orders from the Internet, we believe that the PSP fulfillment infrastructure must fulfill the constant demands of the Internet, while maintaining a capital intensive, but efficient infrastructure. A digital PSP has the twin competitive pressure of being flexible to varied short run job type, and having their equipment run at maximum efficiency. How can we reconcile these competing demands?

This paper posits new innovation of directly connecting the structure of the PSP to the structure of the content that it is seeing. For long runs, we analyze each job as they come in and set up a fixed pipeline for each run and amortize these set-up costs per run. Many linear estimation models based on the Kalman filter are sufficient to forecast production demands in these slow changing environments. For a mixture of short-run content, the pipelines and connections of the processing elements within the PSP may change in response to the aggregate mixture of the jobs and dynamic conditions within the PSP. We believe that neural networks can analyze a digital PSP design because a PSP has naturally adaptive elements that integrate digital and physical infrastructure with human labor force [1].

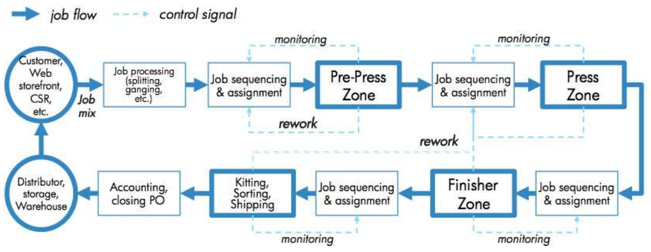

job flow
control signal
monitoring
monitoring
Customer, Web storefront, CSR, etc.
Job mix
Job processing (splitting, ganging, etc.)
Job sequencing & assignment
Pre-Press Zone
Job sequencing & assignment
Press Zone
rework
rework
Distributor, storage, Warehouse
Accounting, closing PO
Kitting, Sorting, Shipping
Job sequencing & assignment
Finisher Zone
Job sequencing & assignment
monitoring
monitoring

Figure 1:A representation of PSP operations as an adaptive network with processing steps, the flow of jobs, monitoring and rework.

To reflect the importance of digital PSP analysis, HP débuted a workflow production software package called SmartStream Production Analyzer. Production Analyzer is a real-time press monitoring software, with the aim to help PSP's reach more effective, more efficient production from Indigo presses. Production Analyzer monitors, tracks and benchmarks the performance of Indigo presses, and aggregates press efficiencies and inefficiencies in terms of press types, shift, and press groups. It provides historical reports of production characteristics (e.g. production performance, up time, printing errors). While software packages like Production Analyzer centralize data collection on PSP operations, we want to go one step further: design PSP based upon on the content they receive.

Representation of PSP Jobs

In this section, we will be talking about the data structure (directed acyclic graphs) that we will be using and how it maps onto the structure and PSP infrastructure (as task graphs) [2].

To formally define the data structure, we will be using some basic graph theory terms. We will be using a subset of graph structures to represent the jobs: directed acyclic graphs (DAG). A DAG is a set of vertices and edges. The vertices in the graph represent progress points of the job fulfillment. The graph is directed: the edge (E) is represented as an ordered pair of vertices (V1, V2), called head and tail, respectively.

$$E = (V_h, V_t); V_h = head(E); V_t = tail(E) \quad (1)$$

An edge represents a processing step in job fulfillment. Let us define a substructure called a path (P) as sequence of edges (of arbitrary length) from the set of the DAG:

$$P = [E_1, E_2, E_3, ..., E_n] \quad (2)$$

such that

$$tail(E_i) = head(E_{i+1}), i = 1,2,...(n-1)$$

(3)

In a directed acyclic graph, there exists no path (P) that has the same beginning and end, i.e. a cycle.

$$\forall P, head(E_1) \neq tail(E_n) \quad (4)$$

While the Directed Acyclic Graph is a general data structure we use, the mapping of a directed acyclic graph onto job going through a processing step is called *a task graph* [Ref to TGFF]. As mentioned before, the vertices represent a state of after processing that has been reached by part or all of the job. The edges represents the processing that is applied during that step to all; the head vertex of the edge represent the processing dependencies that need to be satisfied before the current processing of the edge is to be applied. The tail of the edge represents the change in state after the processing of the edge is applied.

In figure 3, we show a simple task-graph that represents the building of a book. In the case of PSP, a job for a book can be divided into the printing of the book block and the printing of the cover. The book block has to be glued before it is bound together into a single book. The task graph captures all these dependencies while being careful not to enforce a strict ordering between the printing of the book and cover. Either the book block or cover may be printed first, or both may be printed simultaneously. However, before a book can be put together, we must have glued book block and a cover ready. In our particular case, every task graph has a node that represents the start and finish state of the given job. Task graphs are the abstract representation of choice for representing multiple processing steps.

Problem Statement

We characterize the PSP into two passes. In the first pass, we define the input and output of the neural network that will define the structure of PSP; in the second pass, we define the capacity planning cost function with simplifying assumptions, and the output of the capacity planning.

The input and output of the neural network is defined as follows. Input to the neural network is the set of task graphs that represent the jobs that are to be processed by a given structure. The output of the neural network is basic connectivity of the processing elements within the PSP, number of elements in each processing element pools and connectivity between elements, represented by single task graph with capacities on each edge.

Once we have set the structure and connectivity of the PSP; we can then do scheduling optimization with respect to a cost function that contains the types of different processing elements, the cost of each processing elements, their processing and upkeep costs per time unit, and their relative processing times. Also, we simplify the problem in the follow ways:

1. We assume we own all resources from the time of the beginning of the job to the end; cost is for the whole time
2. There are no deadlines on the jobs.
3. We can purchase and fit as many processing units as possible
4. We will not share processing elements across multiple pools

We can, of course, remove these simplifications as will be discussed in our conclusion.

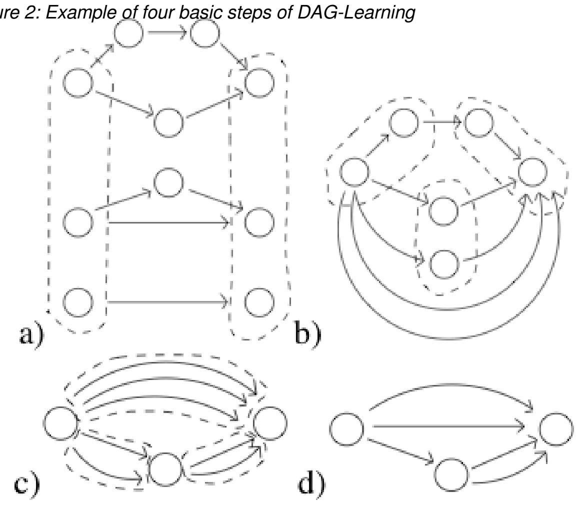
Our Solution: DAGs as Neural Network

We use a four-step technique for the structural reduction of a DAG, a single task graph that represents all jobs to drive the basic structure of the PSP. The steps are 1) the creation of an overall parallel network of things, 2) a scoring function to set the tolerance on how the graphs can be reduced, 3) the identification of exception processes and data error, and 4) the capacity planning with the simplified elements shown above.

As shown in figure 2a, the creation of the initial graph is relatively straightforward: we form a single task-graph formed by joining all the start vertices of each task graph into one global start vertex and all the finish vertices into one global finish vertex. It is straightforward to show that this global graph still maintains all the properties associated with a task-graph. Now that the total information about all jobs is contained in one graph we can work it properly.

Our cost functions are straightforward as well. In the simple case, we match edges by merely matching the processing type associated with the edges. Since the processing methods are all mutually exclusive of each other; this learning operation merely merge the edges by accumulating the number of edges in a counter that go between the same two vertices and degenerates to edges with a count for each edge type. In the general case, we can give a score related to how well two processes match up in a look-up table or analytically. In either case, when we have a well-defined cost function, we can now apply the structural learning algorithms

Figure 2: Example of four basic steps of DAG-Learning



[6] to automatically find redundancies and minimize the size of the total job task graph, as shown in figure 2b,2c, and 2d.

When we receive the structurally reduced job task graph, we can now recognize exceptions by finding the amount of flow through any given part of the sub-graph. We can easily calculate a flow matrix from every vertex to the other things with this simple formula. All edges that are non-zero, but below a given threshold can be flagged exceptions and any task graph original jobs that have these flagged processes can then be flagged as an exceptional job and further analysis can be done to determine whether they are errors or whether they should be considered something else.

Once the connectivity of the PSP is set, we can now work on the capacities of each edge. If we consider the jobs in aggregate, we just take the highest relative strength of the processing unit per each and add one to each while quantizing each unit to the nearest whole number. We can analytically get a static capacity number, or else we can run a simple simulation of for a given optimal scheduling, taking into account pipeline and optimal scheduling. Note that we can expand this problem to include both start time and deadlines and cost function associated with that. We will analyze these issues in our next section how to expand these things. For the general case, we may attempt to a DAG-covering for more optimal capacity planning [5].

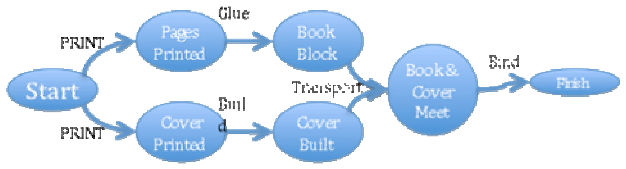


Figure 3: Book Printing Job as Task Graph or DAG; edges are processing, vertices are points of progress

Results

We will present our full results at the NIP26 conference.

Conclusion

PSP are naturally adaptive systems that contain within them human labor and control, and will become even more so when enabled with the deeper integration of digital presses and Internet technologies. The industry has understood the technological impact of digital presses, but has yet to fully appreciate how they can make the PSP infrastructure flexible and agile to match their connectivity to the Internet.

While capital investment demands structural efficiency, we believe that the core value proposition of a PSP is to be flexible and adaptive to internet content that has no inherent model. Here neural networks and their algorithms embody the best of both worlds: a fixed structure that adapts its connectivity to handle the changing inputs robustly.

We have presented a method to obtain the recommended structure and connectivity of the flow of work in a PSP based on the pool of submitted jobs. As previously mentioned, the result is assuming a minimal set of requirements to arrive at a solution under an ideal scenario. While useful in planning a new factory layout, in practice, this output needs to conform to the physical limitation and the optimizing parameters of the PSP at that time to provide some real advantage. The next step consists of narrowing the solution by taking into account the physical limitation such as existing equipment pool and labor force available, and transport of the parts. On another level, the cost associated with job deadlines, equipment set-up and utilization is important to account for in the optimization step. On the connectivity side, we so far assumed that the equipment is used exclusively within a pool. The reality is much more complex. While valid for certain processes, for others such as some types of finishing equipment, the processing elements are shared across multiple pools. Finally, the window of jobs used and the refresh rate also needs to be studied. The initial results are encouraging and the combination of DAG with neural networks provides an interesting platform to solve the duality of high adaptability and high efficiency which is unique to the PSP manufacturing space.

References

[1] E. A. Lee. "Cyber Physical Systems: Design Challenges", International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), May, 2008

[2] Dick, R. P., Rhodes, D. L., and Wolf, W. 1998. TGFF: task graphs for free. In *Proceedings of the 6th international Workshop on Hardware/Software Codesign* , Washington, DC.

[3] "A Novel Learning Method By Structural Reduction Of DAGS For On-Line OCR applications," I-Jong Lin and S.Y. Kung, Proceedings of ICASSP98

[4] Coding and Comparison of DAGs as a Novel Neural Structure with Applications to On-Line Handwriting Recognition," I-Jong Lin, S.Y. Kung, IEEE Trans. on Sig. Proc., Nov. 1997, v. 45, #11, pp. 2701-2708

[5] "MOCSYN: Multiobjective Core-Based Single-Chip System Synthesis", Robert P. Dick and Niraj K. Jha, Proc. Design Automation & Test in Europe Conf, 1999, pg 263-270

Author Biography

I-Jong Lin is the principal research scientist and research manager for GPU-RIP Print Services group with Hewlett-Packard Laboratories. He received PhD and MS in Electrical Engineering from Princeton University and Stanford University, respectively, and BS in Computer Systems Engineering from Stanford University. His research experiences span from VLSI-CAD, digital design, neural networks, video object extraction, to digital print. He is the author of "Video Object Extraction and Representation: Theory and Applications" by Kluwer Academic Publishers. I-Jong holds 13 patents.

Eric Hoarau is a senior researcher with Hewlett-Packard Laboratories (Palo Alto, California). He received his B.S. from the University of California at Berkeley and his M.S. in Mechanical Engineering from the Massachusetts Institute of Technology. His research interests span several fields: Mechatronic systems, imaging systems, color imaging algorithms, distributed computing and system dynamics.

Jun Zeng is a senior scientist with Hewlett-Packard Laboratories. PhD and MS from Johns Hopkins University. Jun's publication includes 40 peer-reviewed papers and a co-edited book on computer aided design of integrated systems. He was a guest editor of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Jun also serves as termed faculty and member of PhD dissertation committee with Duke University's Electrical and Computer Engineering Department.