Computers & Industrial Engineering 108 (2017) 165–176

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A new Hybrid Filtered Beam Search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri Nets

Gonzalo Mejía [a,*], Karen Niño [b]

[a] School of Industrial Engineering, Pontificia Universidad Católica de Valparaíso, Av. Brasil 2241, Valparaíso, Chile
[b] PIT Research Group, Department of Industrial Engineering, Universidad Militar Nueva Granada, Carrera 11 No. 101-80, Bogotá D.C., Colombia

ARTICLE INFO

Article history:
Received 12 April 2016
Received in revised form 28 March 2017
Accepted 18 April 2017
Available online 20 April 2017

Keywords:
Petri Nets
Filtered Beam Search
Deadlock-free scheduling
Flexible manufacturing systems
Beam Search

ABSTRACT

This paper presents a new Hybrid Filtered Beam Search algorithm for deadlock-free scheduling of flexible manufacturing systems. The proposed algorithm uses a diversification/intensification strategy that aims to selectively explore the state space of a Timed Place Petri Net that represents the dynamics of the system. The exploration strategy is based on a double filtering mechanism that limits the number of markings that can be expanded at each level of the search tree. The algorithm, which was tested on several benchmark instances from the literature, not only produces good quality schedules but it is also efficient in terms of memory requirements and computational time. In addition, the algorithm requires little tune-up and produces consistent results. Computational experiments show the effectiveness of the algorithm in comparison with other recent and state of the art methods.

1. Introduction

Petri Nets have been extensively used to model and to schedule flexible manufacturing systems due to their capability to model the asynchronous, concurrent and non-deterministic nature of such systems. Petri Nets can also be used with different optimization methods for scheduling. Recently, the issue of deadlock-free scheduling in combination with Petri Nets has caught the attention of the research community. Deadlock controllers based on Petri Nets can be incorporated into the execution logic and therefore deadlock-free schedules can be calculated and easily verified.

Petri Nets in combination with Informed Graph Search (IGS) algorithms have been very popular for deadlock-free scheduling (Luo, Xing, Zhou, Li, & Wang, 2015). The idea of these methods is to selectively generate the Petri Net state space with the objective of finding a path from one or more initial nodes to a goal node with the minimum path cost. In all IGS algorithms, all generated-but-not-yet-expanded nodes are kept on a dynamic list denoted generically as the "frontier". Nodes on such a list are called "frontier nodes". The way in which the frontier nodes are selected for expansion is perhaps the main difference among all IGS algorithms.

A very well-known IGS algorithm is the A* search algorithm: this algorithm expands the most promising branches of a search tree according to a criterion established with the heuristic function $f(N) = g(N) + h(N)$. The term $g(N)$ is the actual cost from the initial node to a node N and $h(N)$ is an estimate of the cost from such a node to a desired goal node. Those nodes having lower values of $f(N)$ will have priority for expansion. If $h(N)$ does not overestimate the true value of the optimal path cost from N to the goal node, the A* Search algorithm will eventually find an optimal solution. In such a case, the term $h(N)$ is said to be an admissible heuristic function.

One of the major caveats of the A* Search algorithm is its complexity which is exponential in both time and memory requirements (Russell & Norvig, 1995) which make it suitable only for very small instances. If no provisions are taken, the search can quickly degenerate to a Breadth-First Search strategy (Reyes Moro, Yu, Kelleher, & Lloyd, 2002) with only nodes of the lower levels of the search tree being expanded.

The other major caveat of the A* Search algorithm is the calculation of the heuristic function $h(N)$. In theory, having admissible and well informed heuristic functions $h(N)$ will greatly reduce the memory requirements and will speed up the search process. Recall that an admissible function $h_1(N)$ is more informed than $h_0(N)$ if for all nodes of the search tree $h_1(N) \geq h_0(N)$. Finding good admissible heuristic functions is very hard in scheduling problems. In classical scheduling theory, the value of $h(N)$ constitutes a lower bound of the estimated remaining time. For classical flow and job shop scheduling problems, a good lower bound is already hard to calculate (Blażewicz, Domschke, & Pesch, 1996). In the case of

* Corresponding author.
E-mail addresses: gonzalo.mejia@pucv.cl (G. Mejía), karen.nino@unimilitar.edu.co (K. Niño).

http://dx.doi.org/10.1016/j.cie.2017.04.034
0360-8352/

deadlock-prone manufacturing systems, it is even harder: in such systems, resource utilization is generally very poor (Viswanadham, Narahari, & Johnson, 1990) thus creating both resource idle times and job waiting times that can very difficult to estimate.

The first papers that combined Petri Nets and IGS were presented by Lee and DiCesare (1994) who introduced a modified version of the A* Search algorithm. In its original version, the authors attempted to force the search towards the goal node by assigning more weight to those nodes deeper in the search tree to avoid the state space explosion. Later on, other similar algorithms have provided better performance in both quality and speed. In general, most approaches use the A* Search basic algorithm with (i) extensions that limit the nodes to explore and (ii) with new heuristic functions.

Strategies to selectively reduce the number of explored nodes include pruning (Abdallah & ElMaraghy, 1998; Abdallah, ElMaraghy, & ElMekkawy, 2002; ElMekkawy & ElMaraghy, 2003; Tien-Hsiang, Chao-Weng, & Li-Chen, 1994), limiting the amount of backtracking during the search (Huanxin Henry & MengChu, 1998; Jeng & Chen, 1998; Lefebvre, 2016; Lei, Xing, Han, & Gao, 2017), the Dynamic Window Search (DWS) (Luo et al., 2015; Reyes-Moro, Yu, & Kelleher, 2002) which limits the expansion of nodes within a "moving window" of levels, the Reactive A* Search (Kim, Suzuki, & Narikiyo, 2007) which uses a rule-based supervisor to reduce the search space, the hybrid A*-Depth First Search (Huang, Sun, & Sun, 2008; Huang, Sun, Sun, & Zhao, 2009) that forces the search deeper in the reachability graph with a control mechanism, and the Anytime Heuristic Search (Baruwa & Piera, 2014; Baruwa, Piera, & Guasch, 2015) which combines a breadth-first iterative deepening A* Search with breadth-first heuristic search and backtracking.

In terms of heuristic functions, the literature shows that both admissible, non-admissible functions and combinations have been proposed: admissible functions are generally based on structural features of the Petri Net such as minimum path length (Lei, Xing, Han, Xiong, & Ge, 2014; Luo et al., 2015; Mejía & Montoya, 2009; Reyes-Moro et al., 2002), state equations (Jeng & Chen, 1998; Jeng, Chiou, & Wen, 1998; Lee & Lee, 2010; Lei et al., 2014, 2017), resource load (Huang et al., 2008; Huanxin Henry & MengChu, 1998) and on the critical path method (Mejía et al., 2016); non-admissible functions use dispatching rules (Mejía & Odrey, 2005), on resource idle and job wait times (Huang, Jiang, & Zhang, 2014; Luo et al., 2015). Combinations have been also proposed: Some authors (e.g. Mejía et al., 2016) use admissible functions with ties broken with non-admissible functions. The literature shows no definite indication on which, admissible or non-admissible functions have better performance and it seems that the choice is problem dependent.

Another popular IGS is the Beam Search (BS) algorithm which limits the search space by expanding only the best β nodes at each level of the graph of states. The parameter β is called the "beam width". With this strategy, the search progresses in a controlled manner towards the goal node. The algorithm has been effective on a number of combinatorial problems and its running time complexity is polynomial (Ow & Morton, 1988). Its main disadvantage is the node selection which eventually prunes off good nodes from the search tree. Extensions of the BS algorithm are (i) the Filtered Beam Search (Ow & Morton, 1988) which pre-evaluates and filters out nodes with a simpler function before applying a more complex evaluation function to the remaining nodes and (ii) the Recovering Beam Search (RBS) which applies to each node, a further test that checks whether the current node is dominated by another node at the same level (Croce, Ghirardi, & Tadei, 2004). The BS algorithm in combination with Petri Nets has been studied in (Mejía & Montoya, 2009; Mejía & Odrey, 2005; Mejía et al., 2016). All of them combine the expansion strategy of the A* Search with the pruning method of BS.

More recently graph heuristic search methods in combination with deadlock controllers have been studied (Han, Xing, Chen, Lei, & Wang, 2013; Han, Xing, Chen, & Xiong, 2015; Lei et al., 2014, 2017; Luo et al., 2015; Xing, Han, Zhou, & Wang, 2012) for deadlock-prone manufacturing systems. The main advantage of controllers is that they both prevent/avoid deadlock states and limit the search space. The disadvantage is that such controllers can be difficult to derive, generally avoid valid states and cannot be easily generalized to other classes of Petri Nets.

Other approaches use evolutionary algorithms and Petri Nets for scheduling deadlock-prone manufacturing systems: Gang and Wu (2004) proposed a Genetic Algorithm (GA) for FMS in which non-feasible chromosomes are discarded. Later, Dashora, Kumar, Tiwari, and Newman (2008) introduced an evolutionary algorithm along with the widely known supervisors presented by Ezpeleta, Colom, and Martinez (1995) for FMS scheduling. Mejía, Montoya, Cardona, and Castro (2011) proposed a GA with a chromosome which used a transition conflict-breaking mechanism and a rejection strategy for non-feasible solutions. Xing et al. (2012) presented a GA search method which included several common deadlock avoidance supervisors. An improved version of their method was presented later by Han, Xing, Chen, Lei, and Wang (2014) with different crossover and mutation operators. Han, Xing, Chen, and Xiong (2015) proposed a Particle Swarm Optimization (PSO) scheduling method in combination with Petri Net deadlock controllers.

In this paper, we propose a new search strategy that combines a Petri Net-based Beam Search method with a double filtering mechanism for scheduling deadlock-prone flexible manufacturing systems. The proposed algorithm iteratively runs the filtered BS algorithm with an upper bound calculated from previous iterations. The algorithm was tested on a variety of benchmark instances from the literature and the results are analyzed and discussed. The remainder of the paper is organized as follows: Section 2 introduces the concepts and definitions related to Petri Nets. Section 3 describes the Iterated Hybrid Filtered Beam Search approach. Section 4 presents the computer tests, results and analysis. Finally, section 5 discusses conclusions and further research.

2. Background

2.1. Definitions and notation

In this section, we adopted most of the notation used by Ezpeleta et al. (1995) which is pretty much standard on this topic.

Timed Place Petri Net (TPPN): A TPPN ($\mathcal{N}$) is a 6-tuple (P, T, $P \times T$, $T \times P$, W, τ) where P = set places, T = set of transitions, $P \cap T = \varnothing$, $P \times T$ = a set of input arcs to transitions, $T \times P$ = a set of output arcs from transitions, W is the set of arc weights of $P \times T \cup T \times P$, and τ = set of time delays associated with places. This paper deals with ordinary Petri Nets and therefore all arc weights are set to 1, i.e. $w = 1\ \forall w \in W$. The term $\tau(p)$ is the time delay associated to place $p \in P$.

The set $\bullet t$, (resp. $t\bullet$) $t \in T$ contains all input (resp. output) places of transition t. A Petri Net in which $|\bullet t|$ and $|t\bullet| = 1$ is called a state machine. Likewise, the set $\bullet p$, (resp. $p\bullet$) $p \in P$ contains the input (resp. output) transitions of place p. A path in a Petri Net is a sequence of nodes (places or transitions) connected by an arc. Because their own nature, places and transitions alternate in any valid path. A path linking p and p' is denoted as $\chi(p, p')$.

A Marking M is a mapping $P \rightarrow Z$, $Z = [0, 1, 2, 3, \ldots, n]$ where $n = |P|$. $M(p)$ represents the number of tokens at place p. A place p is marked if $M(p) > 0$. The initial distribution of tokens in the net is