Computers and Chemical Engineering 47 (2012) 202–216

Contents lists available at SciVerse ScienceDirect

Computers and Chemical Engineering

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

SmartGantt – An interactive system for generating and updating rescheduling knowledge using relational abstractions

Jorge Palombarini [a], Ernesto Martínez [b,*]

[a] GISIQ (UTN), Av. Universidad 450, Villa María 5900, Argentina
[b] INGAR (CONICET-UTN), Avellaneda 3657, Santa Fe S3002 GJC, Argentina

ARTICLE INFO

Article history:
Received 2 February 2012
Received in revised form 15 June 2012
Accepted 18 June 2012
Available online 30 June 2012

Keywords:
Batch plant management
Cognitive production systems
Manufacturing control
Rescheduling
Relational reinforcement learning
Uncertainty

ABSTRACT

Generating and updating rescheduling knowledge that can be used in real time has become a key issue in reactive scheduling due to the dynamic and uncertain nature of industrial environments and the emergent trend towards cognitive systems in production planning and execution control. Disruptive events have a significant impact on the feasibility of plans and schedules. In this work, the automatic generation and update through learning of rescheduling knowledge using simulated transitions of abstract schedule states is proposed. An industrial example where a current schedule must be repaired in response to unplanned events such as the arrival of a rush order, raw material delay, or an equipment failure which gives rise to the need for rescheduling is discussed. A software prototype (*SmartGantt*) for interactive schedule repair in real-time is presented. Results demonstrate that responsiveness is dramatically improved by using relational reinforcement learning and relational abstractions to develop a repair policy.

1. Introduction

Increasing global competition, a shift from seller markets to buyer markets, mass customization, operational objectives that highlight customer satisfaction and the need to ensure a high level of efficiency in production systems give rise to a complex shop-floor dynamics due to unplanned and disruptive events in industrial environments (Henning & Cerdá, 2000; Zaeh, Reinhart, Ostgathe, Geiger, & Lau, 2010). Moreover, stringent requirements with regard to reactivity, adaptability and traceability in production systems, and by extension in supply chains, are demanded for products and processes by both suppliers and clients all over the product lifecycle.

In order to deal with the above challenges, is necessary to achieve higher degrees of flexibility, adaptability, autonomy and learning capabilities in production systems. Disruption management, self-reconfiguration and adaptive behavior are key capabilities in order to fulfill the aforementioned requirements without sacrificing cost effectiveness, product quality and on-time delivery. In this context, established production planning and control systems are vulnerable to unplanned events and intrinsic variability of a manufacturing environment where difficult-to-predict circumstances occur as soon as schedules are released to the shop-floor. Equipment failures, quality tests demanding reprocessing operations, rush orders, delays in material inputs from previous operations and arrival of new orders give rise to uncertainty in real time schedule execution. Hence, schedules generated under the deterministic assumption are often suboptimal or even infeasible (Li & Ierapetritou, 2008; Vieira, Herrmann, & Lin, 2003; Zaeh et al., 2010). As a result, reactive scheduling is heavily dependent on the capability for generating and representing knowledge about strategies for repair-based scheduling in real-time. Moreover, timely producing satisfactory schedules rather than optimal ones, in reasonable computation time and fully integrated with enterprise resource planning and control systems is mandatory for responsiveness and agility (Trentesaux, 2009).

Existing literature related to reactive scheduling mainly aims to exploit peculiarities of a specific problem structure (Adhitya, Srinivasan, & Karimi, 2007; Miyashita & Sycara, 1994; Miyashita, 2000; Zhu, Bard, & Yu, 2005; Zweben, Davis, Doun, & Deale, 1993). More recently, Li and Ierapetritou (2008) have incorporated uncertainty in the form of a multi-parametric programming approach to generate rescheduling knowledge for specific events. However, the tricky issue is that resorting to a feature-based representation of schedule states is very inefficient, and generalization to unseen schedule states is highly unreliable. Therefore, transferring heuristics or a rescheduling policy is difficult to unseen scheduling domains, being the user-system interactivity severely affected due to the need of compiling the repair-based strategy for each disruptive event separately.

* Corresponding author. Tel.: +54 342 4534451; fax: +54 342 4553439.
E-mail address: ecmarti@santafe-conicet.gob.ar (E. Martínez).

0098-1354/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compchemeng.2012.06.021

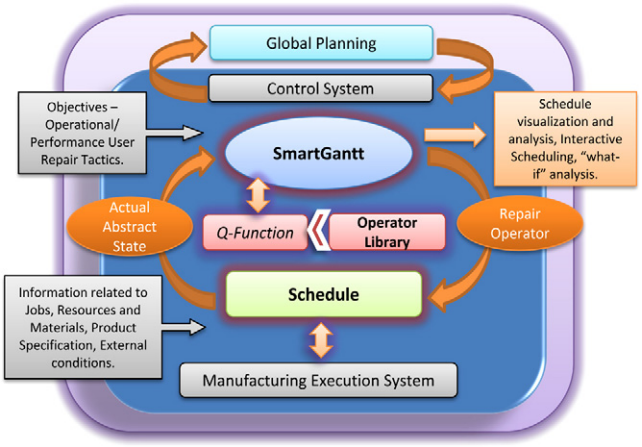

Fig. 1. Repair-based architecture implemented by *SmartGantt*.

Most of the existing works on rescheduling prioritize rescheduling *efficiency* using a mathematical programming approach (Li & Ierapetritou, 2008). However, schedule *stability* is also an important objective that must be accounted for when generating a rescheduling policy (Pfeiffer, Kádár, & Monostori, 2007; Rangsaritratsamee, Ferrell, & Kurz, 2004). Moreover, the type of abstractions used for representing schedule states and repair actions is of paramount importance to scale up solutions found for small-size case studies to industrial applications involving thousands of tasks and hundreds of resources in an uncertain environment. In particular, humans can succeed in rescheduling thousands of tasks and resources by increasingly learning a repair strategy using a natural abstraction of a schedule situation or state: a number of objects (tasks and resources) with attributes and relations (precedence, synchronization, etc.) among them. First-order relational representations enable exploiting the existence of domain objects, of relations (or, properties) over these objects, and make room for quantification over objectives (goals), action effects and abstract properties of schedule states (Blockeel, De Raedt, Jacobs, & Demoen,1999). The very success of Gantt charts (Wilson, 2003) as a support tool for (re)scheduling tasks at the shop-floor level is that they provide a ready visualization of precedence and synchronization relationships between tasks and resources over a common time line.

In this work, a novel real-time rescheduling prototype application called *SmartGantt* based on a relational (deictic) representation of (abstract) schedule states and repair operator is presented. To learn a near-optimal policy for rescheduling using simulations of schedule state transitions (Croonenborghs, 2009), an interactive repair-based strategy bearing in mind different goals and disruptive events is proposed. To this aim, domain-specific knowledge for reactive scheduling is generated and updated using relational reinforcement learning (RRL) (Džeroski, De Raedt, & Driessens, 2001) and relational abstractions (De Raedt, 2008).

2. Repair-based (re)scheduling in *SmartGantt*

Fig. 1 depicts the repair-based architecture implemented by *SmartGantt*, embedded in a more general setting that includes an enterprise resource planning (ERP) system and a manufacturing execution system (MES) along with a communication and control infrastructure. *SmartGantt* also integrates artificial cognitive capabilities in resources and processes through a human–agent–machine interface to favor achieving by design the type of flexibility and adaptability that are needed in production systems (Trentesaux, 2009). The mentioned infrastructure consists of the system control level, the process control level and the planning level. The system control level is responsible for the physical execution of the production process. The global planning level administrates, coordinates and dispatches the incoming orders to the production system, and also involves transport control and planning/scheduling features. Transport control is responsible for the material supply to the production system whereas planning/scheduling capabilities are integrated in *SmartGantt*. The respective order data (e.g. slack time), the current boundary conditions (e.g. machine availability) and the overall system utilization (e.g. capacity utilization) are the prerequisites to decide when an order is released to the shop-floor. Based on the specification of the order and information related to operational objectives, tasks, resources and materials, *SmartGantt* allocates the respective production operations to feasible resources in the production system and the needed resources are booked based on estimated production times. Later on, specific knowledge about resource states (e.g. capability profiles) as well as the respective product-related (e.g. quality) and process-related data (e.g. production steps) are used for feasible schedule modifications (e.g. due to a rework operation) and the on-going repair sequence optimization is applied to face unforeseen events (e.g. machine breakdowns) in an autonomic way.

In *SmartGantt*, rescheduling knowledge about optimal selection of repair operators towards a goal state is generated through reinforcements using a simulator of schedule state transitions. In the simulation environment, an instance of the schedule is interactively modified by the learning system which executes control actions using a sequence of repair operators until a repair goal is achieved. In each learning episode, *SmartGantt* receives information from the current schedule situation or state s, and then selects a repair operator a, which is applied to the current schedule, resulting in a new one. The evaluation of the resulting quality of a schedule after a repair operator has been used by *SmartGantt* is performed using the simulation environment via an objective or reward function $r(s)$. The learning system then updates its action-value function $Q(s,a)$ that estimates the value or utility of resorting to the chosen repair operator a in a given schedule state s. Value updates are made using a reinforcement learning algorithm (Sutton & Barto, 1998), such as the following Q-learning rule

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_b Q(s',b) - Q(s,a)] \quad (1)$$

where s' is the resulting state of using the repair operator a at the schedule state s. Accordingly, a repair operator in a given schedule state is chosen based on the *optimal policy*: $a_i = \pi^*(s_i)$ if a_i has the highest Q-value in the state s_i. A simulation-based algorithm to learn the optimal policy is the well-known Q-learning (Sutton & Barto, 1998; Watkins, 1989).

Based on learning episodes, the Q-learning algorithm updates these Q-values incrementally while the reinforcement learning agent interacts with a real or simulated environment. In this work, the relational variant of Q-learning is used (see next section for details). The main benefit of applying reinforcement learning techniques, such as Q-learning algorithm, in automated generation of rescheduling knowledge for improving stability and efficiency of real-time reactive scheduling is that there is no extra burden on domain experts, allows online adaptation to a dynamic environment and make room for relational abstractions that can be used to deal with large state spaces, e.g. in supply chains or distributed plant scheduling. By accumulating enough experiences over many simulated transitions between schedule states, *SmartGantt* is able to learn an optimal policy for choosing the best repair operator at each schedule state in the path to a goal state (repaired schedule).

For repairing a schedule, *SmartGantt*, is given a repair-based *goal* function $goal{:}S \rightarrow \{true, false\}$ defining which states in the schedule are target states in a repaired schedule, e.g. states where Total