Throughput maximization of capacitated re-entrant lines through fluid relaxation

Michael Ibrahim and Spyros Reveliotis

***Abstract*—This paper extends the scheduling methodology for complex stochastic networks that is based on the solution of a "fluid" relaxation at each decision point of the original scheduling problem, to stochastic networks with blocking and deadlocking effects. For a clearer and more concrete treatment, the presented results are developed in the operational context of a re-entrant line with finite buffering capacity at each workstation; these re-entrant lines are characterized as "capacitated re-entrant lines (CRLs)". From a methodological standpoint, the paper results are enabled by a pre-established ability to control the underlying resource allocation for deadlock freedom, and by the further ability to express the corresponding deadlock avoidance policy as a set of linear inequalities on the system state. Also, the employed "fluid" relaxation for this new regime differs from the "fluid" relaxations that have been employed in past implementations of the method, since it must account for the blocking effects that take place in the considered CRLs. The efficacy the presented scheduling method is assessed through numerical experimentation that compares, for a set of "benchmark" CRLs, the performance of the scheduling policies obtained through this method, to (i) the performance of the corresponding optimal scheduling policies, and also to (ii) the performance of some other heuristic scheduling policies for these systems that are adapted from the relevant literature. Finally, an additional set of experiments demonstrates and assesses the scalability of the presented method by applying it to some pretty large system configurations.**

***Note to Practitioners* – While the real-time management / scheduling of complex resource allocation systems (RAS) is a thriving area in general, the particular problem of scheduling such systems with extensive blocking and deadlocking effects in their operation has received very limited attention. To a large extent, this is due to the fact that the effective scheduling of this particular RAS class requires the initial resolution of an additional problem, of a more combinatorial type, that concerns the establishment of "liveness" for the underlying workflow, i.e., the ability of all the activated jobs to proceed to their completion securing successfully all the required resources for the execution of their various processing stages, and avoiding the formation of any deadlocks or livelocks. On the other hand, this problem of liveness-enforcement for the considered RAS has received extensive attention within a certain part of the controls community during the past decades, and the currently available results provide a broad range of methods and policies for supporting the necessary supervision. This paper combines the aforementioned results on RAS liveness-enforcing supervision with a scheduling methodology that has been very popular in the context of other hard scheduling problems, in order to develop a complete scheduling methodology for the considered RAS. Extensive numerical experimentation reported in the paper demonstrates and assesses the efficacy of the presented method. Finally, in an effort to provide more concrete and simpler exposition for the presented developments, the results are presented in the operational context of a "capacitated re-entrant line", i.e., a particular RAS class modeling the operation of re-entrant lines with finite buffering capacity at its various workstations.**

The authors are with the School of Industrial & Systems Engineering, Georgia Institute of Technology; emails: {michael.ibrahim@,spyros@isye.}gatech.edu.
This work was partially supported by the NSF grant ECCS-1405156.

***Index Terms*—Stochastic scheduling, stochastic networks with blocking and deadlocking effects, complex resource allocation systems, liveness-enforcing supervision, fluid relaxation**

I. Introduction

This paper deals with the problem of the effective and efficient real-time management of complex resource allocation systems (RAS) with blocking and deadlocking effects. This is a particular class of scheduling problems that has received very limited attention by the classical and even the more modern scheduling theory [1], [2], [3]. We believe that a primary reason for this reality is the fact that the effective scheduling of this particular class of systems requires the resolution of an additional type of problem that concerns the establishment of "liveness" for the underlying workflow, i.e., the ability of all the activated jobs to proceed to their completion securing successfully all the required resources for the execution of their various processing stages, and avoiding the formation of any deadlocks or livelocks.

On the other hand, the problem of the coordination of the complex, sequential resource allocation that takes place in many contemporary applications, in order to prevent the formation of the potential deadlocks and livelocks that were mentioned in the previous paragraph, has been studied extensively within the Discrete Event Systems (DES) community, as a particular application of DES Supervisory Control (SC) theory [4], [5]. The currently available results from these studies are very powerful, and they can provide SC policies – known as *Deadlock Avoidance Policies (DAPs)* or *Liveness Enforcing Supervisors (LES)* in the corresponding application settings – that can ensure deadlock freedom and live operation for the target applications while remaining very tractable as these applications scale up with respect to (w.r.t.) their size and their operational complexity. A comprehensive and systematic exposition of the current theory on liveness analysis of complex RAS, and the liveness enforcing supervision of these systems through the effective synthesis and the deployment of the aforementioned DAPs, can be found in [6].

However, as pointed out in the introductory chapter of [6], the currently available SC theory for RAS deadlock avoidance constitutes *preventive* – also known as *"behavioral" or 'logical"* – *control* for the underlying RAS; i.e., the corresponding SC policies essentially seek only to block decisions or actions that could be detrimental for the liveness of this system. Hence, at each decision point of the considered resource allocation function, the DAPs that are offered by the aforementioned theory of [6], act as "filters" that restrain the original set of available decisions / actions to an *admissible* subset, weeding out the elements of the original set that might lead to problematic behavior. But there is a remaining need for additional control logic that will select a particular decision or

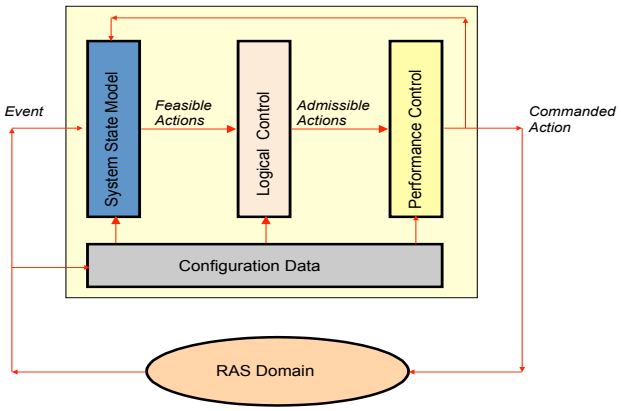

Fig. 1: An event-driven control scheme for the real-time management of the considered RAS [7]. The controller responds to the various events taking place in the controlled RAS by updating a state model that defines the feasible behavior of this system. This behavior is "filtered" through the logical controller in order to obtain the admissible behavior, i.e., the behavior that is consistent with certain specifications imposed on the RAS operation, including the requirements for deadlock-freedom and liveness. Finally, the admissible behavior is processed through the performance-oriented controller in order to select the particular action(s) among the admissible behavior that eventually will be commanded upon the RAS.

action from the admissible set that is defined by the applied DAP, to be commanded eventually on the plant RAS. This additional control logic must observe certain "performance" requirements for the underlying RAS that are typically expressed by some time-based criteria, like the maximization of some notion of throughput, or the control of the congestion that is experienced by the different entities that go through the considered operations. The corresponding control problem is complementary to the RAS logical control problem for deadlock avoidance, and it essentially constitutes a *scheduling* problem for the logically controlled RAS. The DES-based real-time controller for the considered RAS that results from the above decoupling of the corresponding decision-making process into (i) behavioral (or logical) and (ii) performance-oriented control (or scheduling), is depicted schematically in Figure 1, and it is further elaborated in the legend of this figure.

When it comes to the systematic investigation of the performance-control problem of the logically controlled RAS, the currently available results are very limited. In [8], [7], [9] it is shown that this problem can be modeled as a Markov Decision Process (MDP), and the applied liveness-enforcing supervision establishes a communication structure for the underlying state space that renders this MDP analyzable and solvable through some very powerful results and algorithms from the broader MDP theory [10]. But the practical applicability of all these algorithms is substantially limited by the very large sizes of the involved state spaces. In fact, the size of these state spaces, when combined with their discrete nature, further implies that even the mere enumeration of an optimal (deterministic) scheduling policy for the considered MDPs is an intractable task, since this enumeration must specify an optimal action for every single state.

In view of the aforementioned complications, in some past works we have tried to address the considered RAS scheduling problem by adapting to it ideas and techniques that come from the burgeoning area of Approximate Dynamic Programming (ADP) [11]. Thus, in [8], [12] we have pursued the idea of developing a "feature"-based approximation of the relative value function of the underlying MDP, along the corresponding theory that is discussed in [13]. But the selection of a good set of "features" – or, more formally, a set of "basis" functions for supporting the sought approximation – is currently an *ad hoc* process, and a naive definition of these features, based on some elementary concepts that are provided by the corresponding queueing and scheduling disciplines, does not enable an effective, controllable trade-off between the computational complexity of the derived policies and their attained performance. Hence, more recently, in [9], [14], [15], we have sought to circumvent this difficulty by pursuing an ADP approximation method that is known as "approximation in the policy space" [11]. Under this new approach, the selection of the sought scheduling policy is optimized over a predefined policy space that (a) admits a parsimonious representation, and (b) is naturally motivated by the underlying problem structure and the prevailing industrial practice. But this approach implies that the quality of the obtained policy is drastically (pre-)determined by the selected policy space. Additional challenges for this method arise from (i) the fact that it uses stochastic approximation [16] for the selection of an optimal parameter setting for the sought policy, which is a computationally intensive proposition, and (ii) the possibility of entrapment in local optima during the policy optimization process.

This paper complements and extends the aforementioned endeavors by introducing a third approach for determining a near-optimal scheduling policy for the considered RAS and the corresponding MDP formulation. This new approach seeks to overcome the limitations of the previously pursued methods, that were discussed in the previous paragraph, by adapting to the considered problem setting an alternative scheduling method that has been employed with extensive success by the OR and IE communities for other classes of complex scheduling problems, that do not present, however, the behavioral problems of deadlock and livelock that are at the core of the resource allocation functions considered in this work; some specific examples of such earlier implementations can be found in [17], [18], [19], [20]. From a methodological standpoint, this new approach will resolve the performance-control problem that is addressed at each decision epoch by the real-time control framework of Figure 1, by

1) first defining and solving a linear programming (LP) formulation that is known as the corresponding *"(fluid) LP relaxation"* in the relevant terminology, and
2) subsequently utilizing the obtained optimal solution for this LP in order to define a selection criterion among the set of decisions / actions that are admissible by the applied DAP at the current decision point.

The aforementioned LP relaxation to be solved at each decision epoch is determined by (i) the structure of the underlying RAS, (ii) the RAS state at the current decision epoch, and (iiii) the control logic of the applied DAP. The

last dependency further implies that the applied DAP must be expressible as a set of linear inequalities on the RAS state; such DAPs are characterized as "linear" in the corresponding literature, and the theory presented in [6] can provide effective and parsimonious realizations of these policies for any given instance from the RAS class(es) to be considered in this work. On the other hand, the considered paper provides

1) a complete definition of the LP relaxation to be formulated and solved at each decision epoch in the context of the considered RAS;
2) the necessary logic for converting the obtained optimal solution of the LP relaxation to specific criteria that will drive the selection of the admissible actions to be commanded upon the underlying RAS at these decision epochs; and
3) extensive numerical experimentation that (i) establishes the ability of the presented method to provide very efficient scheduling policies, while (ii) remaining tractable under the stringent time budgets that must be typically observed in the real-time operational settings of the underlying applications.

In an effort to concretize the presented results and control the expository complexity of the subsequent developments, in the following we shall restrict our consideration to the particular RAS class of the *capacitated re-entrant line (CRL)*, seeking to develop a detailed scheduling methodology that will maximize the long-term throughput of any given CRL instantiation. This choice is also in line with a similar practice that was adopted in the aforementioned developments of [8], [12], [9], [14], [15],

We remind the reader that, in the context of the existing scheduling literature, the classical (uncapacitated) re-entrant line (RL) essentially constitutes a "linear" (i.e., strictly sequential) workflow where some of the supported processing stages share the same workstation [21]. Furthermore, this classical re-entrant line has been the object of extensive study by the past scheduling theory since it is one of the simplest workflow layouts that gives rise to many of the challenging scheduling problems that are also encountered in more complex operations.[1] Hence, the current literature avails of some very strong analytical results characterizing and assessing the stability of any instantiation of the uncapacitated RL model under any given scheduling policy [23], [24], [25], [26], [27], and it also provides some very efficient policies that can ensure stable operation while retaining the operational simplicity of the dispatching rules that are frequently used in the relevant production environments [28], [29].

On the other hand, the CRL model that is considered in this work modifies the original RL model by imposing a finite buffering capacity at each of the line workstations. In [30], [31], it is shown that the blocking and deadlocking effects that are introduced by this modification in the underlying workflow dynamics, negate the applicability to the CRL model of the aforementioned results of scheduling theory for the original RL model, and define a need for new scheduling methodologies for this model, like the ones that are pursued in this work.

Finally, in the context of the RAS models and their LES theory that are presented in [6], a CRL constitutes a RAS where the primary administered resources are the finite buffering capacities of the line workstations.[2] Hence, for the purposes of deadlock avoidance, this CRL can be abstracted to a Linear, Single-Unit (L-SU) RAS in the corresponding RAS taxonomy that is presented in [6], with the additional restriction that this RAS supports only one process type. From a more practical standpoint, the CRL classification as an L-SU RAS specifies further the theory that is available for its liveness enforcing supervision, while the fact that it supports only a single process type is very useful for simplifying the subsequent developments since it enables a straightforward definition of the notion of "throughput maximization" that is pursued in the later parts of this document.[3]

In view of the above positioning of the paper content and its intended contribution, the rest of it is organized as follows: Section II introduces the considered CRL model and the corresponding throughput-maximization problem. This section also reviews the main results from the RAS SC theory of liveness-enforcing supervision that are necessary for the complete definition of the addressed CRL scheduling problem, and expresses this scheduling problem as an average-reward continuous-time MDP. A small but elucidating example distributed across the various parts of this section highlights and concretizes the different concepts and structures that are introduced in these parts. Section III presents the scheduling methodology that is pursued in this work, and demonstrates its application through the example problem instance that was introduced in Section II. On the other hand, the efficacy of the presented scheduling method, in terms of the quality of the derived schedules and its scalability, is demonstrated and assessed more thoroughly in Section IV. More specifically, this section reports a series of numerical experiments that implement the presented method on a number of CRL configurations taken from some respective experiments that are reported in [15], and enable us to assess the efficacy of the method in terms of the quality of the derived solutions. At the same time, a complementary set of experiments on larger CRL configurations also demonstrates the scalability of the presented method in terms of the involved computations. Section V concludes the paper, summarizing its developments, highlighting further their broader significance in the context of the existing literature and for the underlying practice, and pointing out some directions for future work. Finally, we also notice, for completeness, that an abridged version of this work

[1] In addition, from a more practical standpoint, the uncapacitated "re-entrant line" model has been extensively promoted as a pertinent abstraction for the representation of the basic workflow that is encountered in many semiconductor manufacturing fabs [22].

[2] Strictly speaking, the workstation servers are additional resources that must be allocated to the running processes for the execution of their various processing stages. But the local nature of these allocations, in terms of the corresponding workstations, implies that they cannot be a source for deadlock or livelock; hence, they are ignored when it comes to the RAS logical control problem of liveness-enforcing supervision. On the other hand, the server allocations are central in the performance-oriented control of these models.

[3] In RAS that support more than one process types, throughput-maximization problems must be addressed under additional constraints that relativize / distribute the attained throughput along the various supported process types; while practically relevant, the inclusion of this additional feature in the subsequent discussion would complicate the exposition of the underlying models and formulations without adding anything substantial to the pursued ideas and methods.

has been accepted for presentation at ACC 2018.[4]

II. The capacitated re-entrant line and the corresponding throughput-maximization problem

This section introduces the considered CRL model, and provides the necessary background material for a detailed characterization of (i) the corresponding logical control problem of deadlock avoidance, and (ii) the complementary scheduling problem of Figure 1; this last problem is posed as the throughput-maximization of the considered CRLs, and it is eventually formulated as an average-reward MDP. The section also highlights / provides pointers to the key results for the synthesis of linear DAPs for Linear Single-Unit RAS, that are necessary in the context of this work.

A. The considered CRL model

Since the considered CRL model can be perceived as a linear, single-unit RAS with a single process type, its basic structure and functionality can be defined in terms of the structure and functionality that characterizes the broader RAS abstraction in the corresponding literature [6].

Hence, in the CRL case, the primary resource types are L single-server workstations, $W_1, W_2, \ldots, W_L$, each possessing finite buffering capacity B_i, $i = 1, \ldots, L$. On the other hand, the process type that is supported by this line is defined by a sequence of M processing stages, $J_1, J_2, \ldots, J_M$. Each processing stage J_j is carried out at one of the line workstations and it requires a slot of the station buffering capacity during its entire sojourn in it. This station will be denoted by $W(J_j)$, and the function $W(\cdot)$ constitutes the resource allocation function for the corresponding L-SU RAS. Furthermore, it is assumed that $L < M$, an assumption that manifests the re-entrant nature of the line.

Some additional assumptions that detail the line operation, are as follows:

A part visiting the workstation $W(J_j)$ for the processing of the corresponding processing stage J_j will receive service from the station server by having the server visiting the buffer slot that accommodates this part. Hence, any part visiting this workstation will remain in its allocated slot during its entire sojourn at the station, and at any time point during this sojourn, the part will either be waiting for processing, be in processing, or will have completed processing and it will be waiting for transfer to the next required workstation.

Furthermore, in line with the corresponding resource allocation theory, a part that has completed the processing of stage J_j, can move to the next required workstation $W(J_{j+1})$ for the execution of its next processing stage, only when there is an available buffer slot at this workstation. Hence, the processed parts are subjected to blocking effects, and when combined with the re-entrant nature of the considered workflow, these blocking effects can also give rise to deadlocks.[5]

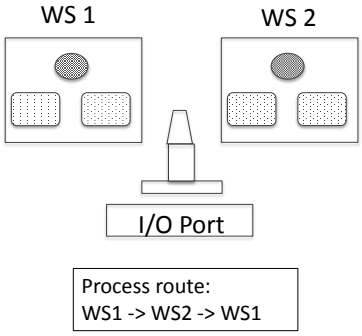

Fig. 2: An example CRL.

The processing times for the processing stages $J_j,\ j = 1, \ldots, M$, are assumed to be exponentially distributed with mean processing time τ_j. And we shall also set $\mu_j \equiv 1/\tau_j,\ \forall j$.[6] Furthermore, part loading and transfer times between the line workstations are assumed to be negligible w.r.t. processing times.[7]

Finally, since in the following developments our primary objective is the throughput maximization of the considered CRL model, we also assume the existence of an "infinite backlog" of parts waiting for processing in front of the line; i.e., the line never starves for work.

Example: We concretize the definition of the CRL model that was provided in the previous paragraphs, through the example manufacturing system that is depicted in Figure 2. This system consists of two workstations, labeled as WS_1 and WS_2 in Figure 2, and an I/O port that interfaces it with the rest of its operational environment. Each workstation has a single server, depicted as a grey ellipse in Figure 2, and two buffer slots, depicted by the corresponding rectangles. Parts visiting each of the two workstations are accommodated at one of the available buffer slots, and they are processed by the workstation server by having the server visiting the corresponding slot. A robotic manipulator supports the necessary material handling functions, and integrates the entire facility to a fully automated cell. Figure 2 also provides the process route for the parts that are processed through this CRL; since workstation WS_1 is visited twice by each part, the considered layout constitutes a re-entrant line. Furthermore, letting $J_j,\ j = 1, 2, 3$, denote the three processing stages of this CRL,

[4]The ACC 2018 write-up constitutes a much more concise development of Sections II and III, and includes neither the experimental results that are reported in Section IV, nor the expansive introductory discussion of this work that connects the presented results to the RAS real-time control problem of Figure 1.

[5]We also want to notice that while the adopted service model for the line workstations intends to provide a concrete base for the exposition of the presented developments, it is not restrictive in any strong sense, since the presented method can be easily adapted to other service models that are employed by such workstations. From a more practical standpoint, this service model for the line workstations is a quite faithful abstraction of the workflow that is materialized at the various chambers of the, so called, "cluster tools" [32], that constitute a prevailing technology in the current semiconductor manufacturing.

[6]While the assumption of exponential processing times is meant to simplify the exposition of the theory that is developed in this paper, more generally distributed processing times can be handled by approximating them by phase-type distributions to any desired degree of accuracy; please, c.f. to [33] for an introduction to phase-type distributions, and to [5] for a brief introduction on the modeling of non-Markovian dynamics by phase-type distributions.

[7]Non-zero loading and transfer times can be included easily in the considered model through the addition of further stages in the underlying process plan.

under the notation that was introduced in the earlier parts of this section we shall also have $W(J_1) = W(J_3) = WS_1$ and $W(J_2) = WS_2$. Finally, we assume that the processing times for each of the three processing stages are exponentially distributed with corresponding instantaneous rates μ_j and corresponding expected values $\tau_j = 1/\mu_j,\ j = 1, 2, 3$.

B. Abstracting the CRL "untimed" dynamics through a finite state automaton

Following the relevant theory presented in [6], we model the basic structure of the workflow dynamics of the introduced CRL model, and the corresponding resource allocation function, by means of a finite state automaton (FSA) $\Phi \equiv (S, E, f, \mathbf{s}_0, S_M)$. Next, we define the various elements of this automaton Φ.

State and state space: A pertinent definition of a notion of *state* for this automaton can be based on the number of the parts waiting for processing, being processed or having completed processing of the different processing stages, J_j, of the supported process type. More specifically:

Definition 1: The state $\mathbf{s}$ of the CRL model considered in this work is a $3M$-dimensional vector with component $3j + k,\ j = 0, \ldots, M-1,\ k = 1, 2, 3$, denoting respectively the number of parts that are waiting for processing, executing, or having completed processing stage J_j. □

The state set S of the aforementioned automaton Φ consists of all vectors $\mathbf{s}$ that admit an interpretation according to Definition 1, and are compatible with (i) the single-server assumption for the line workstations, and (ii) the available buffering and capacities $B_i,\ i = 1, \ldots, L$, at these workstations. Then, the finite buffering capacity of all workstations implies that the resulting state set S of Φ is, indeed, finite.

Furthermore, since, in the considered CRL model, (a) part loading and unloading require zero time, and (b) there is an infinite backlog of jobs waiting for processing, it is possible to simplify the state concept introduced in Definition 1 by dropping the first and the last component of the state vector $\mathbf{s}$; i.e., parts seeking to execute their first processing stage can be loaded into the line only when the corresponding server is available, and parts having completed processing of the last processing stage can be unloaded immediately. In the following, we shall adopt this simplified state model, with the necessary adjustments in the corresponding notation.

Example: The (simplified) state $\mathbf{s}$ of the FSA Φ corresponding to the CRL of Figure 2 is a 7-dim integer vector. The first two components of this vector $\mathbf{s}$, $\mathbf{s}_1$ and $\mathbf{s}_2$, report, respectively, the number of parts at workstation WS_1 executing processing stage J_1 and having completed the processing of this stage; components $\mathbf{s}_3,\ \mathbf{s}_4$ and $\mathbf{s}_5$ report the number of parts in workstation WS_2 that are, respectively, waiting for the execution of stage J_2, executing this stage, and having completed execution of this stage; finally, components $\mathbf{s}_6$ and $\mathbf{s}_7$ of state $\mathbf{s}$ report the number of parts at workstation WS_1 respectively waiting for the execution of stage J_3 and executing this stage. □

Events and their controllability: The set of *events*, E, that advance state $\mathbf{s}$, consists of (i) the event e^l that loads a new part on the line; (ii) the events $e^a_j,\ j = 1, \ldots, M-1$, that advance a part from workstation $W(J_j)$ to the next requested workstation, $W(J_{j+1})$, allocating to this part a free buffer slot of the new workstation; (iii) the events $e^p_j,\ j = 1, \ldots, M$, that initiate the processing of a part at workstation $W(J_j)$ by allocating to it the corresponding server; (iv) the events $e^d_j,\ j = 1, \ldots, M$, that de-allocate the server upon completion of the part processing; and (v) the event e^u that unloads a completed part from the line.

Furthermore, for the needs of the subsequent developments, it is also pertinent to distinguish the various event types that were defined in the previous paragraph into "controllable" and "uncontrollable" events. More specifically, the events of type (i), (ii), (iii) and (v) are *controllable* by the line supervisor. Furthermore, under the aforestated assumptions, these events are executed in zero time when commanded by the supervisor. On the other hand, the events of type (iv) occur spontaneously upon the completion of the processing of the corresponding part, and therefore, they will be treated as *uncontrollable* events. The reader should also notice that, in the *timed* dynamics of the considered CRL, there is a nonzero lag between a type (iii) event and the execution of the corresponding type (iv) event, that corresponds to the necessary processing time.

The state transition function: The state transition function $f : S \times E$ of automaton Φ is a *partial* function encoding the evolution of the system state $\mathbf{s}$ upon the execution of the different events $e \in E$. In particular, function f is defined only on those pairs $(\mathbf{s}, e) \in S \times E$ where the considered event e is feasible in the corresponding state $\mathbf{s}$. Furthermore, f extends on $S \times E^*$ in the natural manner.[8]

Initial and marked states: For the initial state $\mathbf{s}_0$ of FSA Φ, we set $\mathbf{s}_0 = \mathbf{0}$, i.e., the state where the line is empty of any parts. We also set $S_M = \{\mathbf{s}_0\}$, signifying the fact that an accepting run of FSA Φ should complete all the activated jobs and bring the system back to its initial state.

Deadlock and the need for deadlock avoidance: As remarked in the introductory section, the ability of the considered CRL to reach its marked state $\mathbf{s}_0$ can be compromised by the formation of *deadlock*. In the abstracting representational framework of FSA Φ, deadlock is formally defined as follows:

Definition 2: A CRL *deadlock* is a state $\mathbf{s}$ of the corresponding FSA Φ where there is a subset $\mathcal{I} \subseteq \{1, \ldots, L\}$ such that (i) each workstation $W_i,\ i \in \mathcal{I}$, has its buffer slots fully allocated, and (ii) each part p accommodated in the workstation subset that is defined by the index set $\mathcal{I}$ requests transfer to another workstation in this subset. □

Example: In the FSA Φ that corresponds to the CRL of Figure 2, any state $\mathbf{s}$ that has (a) the buffer slots of workstation W_1 fully allocated to parts executing or having completed their first processing stage, and (b) the buffer slots of workstation W_2 also fully allocated (obviously to parts waiting for the execution / executing / having completed the execution of their second processing stage), is a deadlock. Formally, these deadlock states are represented by the set

$$S^d \equiv \{\mathbf{s} \in S : \mathbf{s}_1 + \mathbf{s}_2 = 2 \ \wedge \ \mathbf{s}_3 + \mathbf{s}_4 + \mathbf{s}_5 = 2\} \quad (1)$$

The reader can also check that the states contained in the above set S^d are the only deadlock states of the CRL considered in this example. □

[8]We remind the reader that, in the relevant automata theory, E^* denotes the *Kleene closure* of the event set E; i.e., E^* contains all the finite-length sequences σ of the elements of E, including the empty sequence ϵ.

It is clear that under the operational assumptions that were stated in Section II-A, parts that are involved in a deadlock formation will be permanently stalled in their current workstations, and at the same time, they will prevent the advancement of any further parts through these workstations. Hence, CRL states containing such deadlock formations must be proactively identified and blocked during the line operation. This task necessitates the deployment of an additional control function on the considered CRL that takes the form of a pertinent *deadlock avoidance policy (DAP)*. As mentioned in the introductory section, the design of a pertinent DAP for any given CRL configuration is supported by the existing supervisory control theory for complex resource allocation systems [6]. Next, we overview some basic developments in this theory that are particularly relevant to the needs of this work; the reader is referred to [6] for a much more expansive exposition of this material.

C. Establishing deadlock freedom for the considered CRL model

An alternative, more compressed representation of the underlying CRL dynamics: It is clear from Definition 2 and its accompanying example that CRL deadlock is due only to the allocation of the workstation buffering capacity, and not to the allocation of the processing capacity of the line servers. Hence, the corresponding problem of deadlock avoidance can be focused on this particular allocation. This can be achieved by considering the further abstraction of the FSA Φ, that was introduced in the previous subsection, to the FSA $\hat{\Phi} = (\hat{S}, \hat{E}, \hat{f}, \hat{\mathbf{s}}_0, \hat{S}_M)$, with a (vector) state $\hat{\mathbf{s}}$ that considers collectively all the parts located at workstation $W(J_j),\ j = 1, \ldots, M$, for the execution of the corresponding processing stage J_j; in other words, each component of the new state $\hat{\mathbf{s}}$ will report the number of parts located at some workstation $W(J_j),\ j = 1, \ldots, M$, for the execution of the corresponding processing stage J_j, without discriminating whether these parts are waiting for processing, are in processing, or have completed processing of this stage and are waiting for transfer to the next required workstation. Furthermore, the event set $\hat{E}$ of $\hat{\Phi}$ will consist only of the type (i), type (ii) and type (v) events of the original FSA Φ. Finally, we also set $\hat{\mathbf{s}}_0 = \mathbf{0}$, and $\hat{S}_M = \{\hat{\mathbf{s}}_0\}$.

Example: For the example CRL of Figure 2, the corresponding FSA $\hat{\Phi}$ has a 3-dim state $\hat{\mathbf{s}}$. Furthermore, for any state $\mathbf{s}$ of the original FSA Φ that was defined in Section II-B, the corresponding state $\hat{\mathbf{s}}$ is obtained through the following equations:

$$\begin{aligned} \hat{\mathbf{s}}_1 &\equiv \mathbf{s}_1 + \mathbf{s}_2 \\ \hat{\mathbf{s}}_2 &\equiv \mathbf{s}_3 + \mathbf{s}_4 + \mathbf{s}_5 \\ \hat{\mathbf{s}}_3 &\equiv \mathbf{s}_6 + \mathbf{s}_7 \end{aligned} \tag{2}$$

Clearly, state $\hat{\mathbf{s}}$ changes only when a part enters or leaves one of the line workstations, and it ignores completely the server allocation at these workstations, as well as the specific processing status of the various parts that are located at these workstations.

It is also interesting to notice that, according to Equations 1 and 2, in the more abstracted representation of the CRL dynamics that is provided by FSA $\hat{\Phi}$, all deadlock formations taking place in the CRL of Figure 2 are represented by the single state $\hat{\mathbf{s}}^d = (2, 2, 0)$. It is this representational compression attained by FSA $\hat{\Phi}$ that renders it useful in the analysis of the corresponding deadlock avoidance problem and in the subsequent developments. □

State reachability, safety and maximally permissive deadlock avoidance: In the notational semantics that are associated with FSA $\hat{\Phi}$, we shall further denote by $\hat{S}_r$ the set of *reachable* states of $\hat{\Phi}$, i.e., the states $\hat{\mathbf{s}} \in \hat{S}$ that are accessible from state $\hat{\mathbf{s}}_0$ through some feasible event sequence $\sigma \in \hat{E}^*$. On the other hand, state set $\hat{S}_s$ will denote the set of *co-reachable* – or *"safe"* – states of $\hat{\Phi}$, i.e., the states $\hat{\mathbf{s}} \in \hat{S}$ from which state $\hat{\mathbf{s}}_0$ is accessible through some feasible event sequence $\sigma' \in \hat{E}^*$. We shall also set $\hat{S}_{\bar{r}} \equiv \hat{S} \setminus \hat{S}_r$ and $\hat{S}_{\bar{s}} \equiv \hat{S} \setminus \hat{S}_s$, and we shall refer to these two sets, respectively, as the sets of the *unreachable* and the *unsafe* states. Finally, we shall also use the notation $\hat{S}_{xy} \equiv \hat{S}_x \cap \hat{S}_y$, for $x \in \{r, \bar{r}\}$ and $y \in \{s, \bar{s}\}$.

It should be clear from the definition of the set $\hat{S}_{rs}$ in the previous paragraph that it comprises all the reachable states $\mathbf{s} \in \hat{S}_r$ for which there exist feasible event sequences, $\sigma \in E^*$, leading to the completion of all the parts that are in execution in these states. In the *state transition diagram (STD)* $\hat{\mathcal{G}}$ representing the dynamics of FSA $\hat{\Phi}$, this property of $\hat{S}_{rs}$ is manifested by the fact that the subgraph induced by its states is the maximal strongly connected component of $\hat{\mathcal{G}}$ containing the empty state $\hat{\mathbf{s}}_0$. These remarks subsequently imply the following characterization of the *maximally permissive DAP* for the considered CRL model:

Theorem 1: In the representational semantics of FSA $\hat{\Phi}$, deadlock can be avoided while imposing the minimal possible restriction on the workflow dynamics of the underlying CRL, by identifying and blocking attempted transitions from subspace $\hat{S}_{rs}$ to subspace $\hat{S}_{r\bar{s}}$. The resulting DAP is characterized as *maximally permissive* in the corresponding literature, it is uniquely defined, and, in the following, it will be denoted by Δ^*. □

Theorem 1 is a specialization to the considered CRL model of some broader developments of [6] concerning the characterization of maximally permissive deadlock avoidance in complex resource allocation systems. In fact, the work of [6] provides also a complete methodology for the effective deployment of the optimal DAP Δ^* for any instantiation of the CRL model that is considered in this work.

Furthermore, the works of [6], [34] present an additional set of results which establish that for a very large subclass – in fact, the majority of the practical instantiations – of the considered CRL model, the optimal DAP Δ^* admits a representation as a set of linear inequalities on the state $\hat{\mathbf{s}}$.

Example: For the example CRL of Figure 2, the reader can check that

$$\hat{S}_{\bar{s}} = \hat{S}_d = \{(2, 2, 0)\} \tag{3}$$

Hence, for this simple CRL, deadlock can be effectively avoided by enforcing the constraint

$$\hat{\mathbf{s}}_1 + \hat{\mathbf{s}}_2 \leq 3 \tag{4}$$

in underlying workflow dynamics. Furthermore, this constraint attains deadlock freedom for the line operation in a maximally permissive manner, since, starting from the initial state $\hat{\mathbf{s}}_0$, the

only reachable state that violates this inequality is the deadlock state $\hat{\mathbf{s}}^d = (2, 2, 0)$. □

Correct linear DAPs and policy "lifting" to FSA Φ: As we shall see in the next section, the ability to represent the employed DAP Δ through a set of linear inequalities on the state $\hat{\mathbf{s}}$ is instrumental for developing the LP relaxation and the corresponding scheduling method that are pursued in this work. In order to address CRL instances where the corresponding maximally permissive DAP does not admit a representation as a set of linear inequalities on state $\hat{\mathbf{s}}$, we also introduce the broader concept of a *"correct linear DAP"*:

Definition 3: A set of linear inequalities imposed on the state $\hat{\mathbf{s}}$ of any given instantiation of the considered CRL model defines a *correct linear* DAP Δ for this CRL if and only if the set $\hat{S}_a(\Delta) \subseteq \hat{S}_r$ containing the reachable states that satisfy these inequalities, induces a strongly connected component, $\hat{\mathcal{G}}_a(\Delta)$, of the corresponding STD $\hat{\mathcal{G}}$, that contains the initial state $\hat{\mathbf{s}}_0$.

Also, the aforementioned state set $\hat{S}_a(\Delta)$ that is induced by a correct linear DAP Δ, is characterized as the (reachable) state (sub-)space that is *admissible* by this policy.[9] □

The developments of [6] also enable the computation of efficient approximations of the optimal DAP Δ^* that take the form of a correct linear DAP Δ, when the optimal DAP Δ^* does not admit a linear representation.

Furthermore, the eventually employed DAP Δ can be "lifted" to the original FSA Φ, that models more completely the operation of the underlying CRL, through a state admission rule that will admit a state $\mathbf{s} \in S$ if and only if (*iff*) the corresponding state $\hat{\mathbf{s}}$ belongs in $\hat{S}_a(\Delta)$. The resulting admissible subspace of S will be denoted by $S_a(\Delta)$, and the subgraph $\mathcal{G}_a(\Delta)$ induced by the state set $S_a(\Delta)$ in the STD $\mathcal{G}$ of the FSA Φ has similar connectivity properties to the connectivity properties of the subgraph $\hat{\mathcal{G}}_a(\Delta)$ w.r.t. the STD $\hat{\mathcal{G}}$. Furthermore, the notions of "(state) reachability" and "co-reachability / safety" are naturally extended to the CRL dynamics that are described by the FSA Φ.

Example: Figure 3 depicts the reachable and safe state space, S_{rs}, for the example CRL of Figure 2. S_{rs} is also the subspace admitted by the maximally permissive, correct, linear DAP, Δ^*, that is defined by Eq. 4. A complete characterization of the various states depicted in this figure is provided in Table I. In the next section, we shall show how to formulate the scheduling problem of the throughput maximization for this CRL as an MDP, by introducing additional information to the STD of Figure 3 that pertains to the "timed" dynamics of the considered CRL.

D. The problem of throughput maximization of the considered CRL model and the corresponding MDP formulation

Introducing "timed" dynamics to FSA Φ – tangible and vanishing states: In this subsection we consider the problem of maximizing the throughput of the considered CRL model, under the supervision of a correct DAP Δ. To fully characterize this problem, and proceed with the corresponding MDP formulation of it, we must augment the FSA-based representation of the workflow dynamics of the considered CRLs with time-related elements. An effective way to perform this augmentation is by differentiating the states $\mathbf{s}$ in the Δ-admissible state space S_a [10] into (a) states where the only enabled events are some uncontrollable events e_j^d, and (b) states that enable controllable events as well,[11] according to the following definition:

Definition 4: Consider a CRL instance controlled by a correct DAP Δ. Then, a state $\mathbf{s} \in S_a$ is characterized as *tangible iff* the only enabled events in $\mathbf{s}$ are some uncontrollable events e_j^d; otherwise, state $\mathbf{s}$ will be characterized as *vanishing*. Furthermore, the entire set of tangible states will be denoted by S_a^T, and the set of vanishing states will be denoted by S_a^V. □

We demonstrate the concepts of "tangible" and "vanishing" states that were introduced in the previous definition, and highlight the significance of these concepts for the subsequent developments, through the following example.

Example: In the example STD of Figure 3, tangible states are depicted as double-circled, while vanishing states are single-circled.

Next, let us focus on tangible state #7. From Table I, it is clear that this state contains two parts: a part p_1 executing

TABLE I: The state description for the STD of Figure 3.

$\mathbf{s}$	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$	$\mathbf{s}$	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$
0	0 0	0 0 0	0 0	33	0 0	0 0 1	1 0
1	1 0	0 0 0	0 0	34	0 0	0 0 0	2 0
2	0 1	0 0 0	0 0	35	1 0	1 0 1	1 0
3	1 1	0 0 0	0 0	36	0 0	0 1 1	1 0
4	0 0	1 0 0	0 0	37	0 0	1 0 0	2 0
5	1 0	1 0 0	0 0	38	0 0	0 1 0	2 0
6	0 0	0 1 0	0 0	39	1 0	0 1 1	1 0
7	1 0	0 1 0	0 0	40	0 1	0 1 1	1 0
8	0 1	0 1 0	0 0	41	1 0	0 0 2	1 0
9	1 0	0 0 1	0 0	42	0 1	0 0 2	1 0
10	1 0	0 0 0	1 0	43	0 1	0 0 2	0 1
11	0 1	0 0 0	1 0	44	0 1	0 0 2	0 0
12	0 0	1 0 0	1 0	45	0 1	0 1 1	0 1
13	0 1	0 0 0	0 1	46	0 1	0 1 1	0 0
14	0 0	1 0 0	0 1	47	0 0	1 1 0	1 0
15	0 0	0 1 0	0 1	48	0 1	0 1 0	0 1
16	0 0	0 0 1	0 1	49	0 0	1 1 0	0 1
17	0 0	0 0 0	1 1	50	0 0	1 1 0	0 0
18	0 0	0 0 0	1 0	51	1 0	1 1 0	0 0
19	0 0	0 0 0	0 1	52	0 1	1 1 0	0 0
20	1 0	1 0 0	1 0	53	1 0	1 0 1	0 0
21	0 0	0 1 0	1 0	54	1 0	0 1 1	0 0
22	1 0	0 1 0	1 0	55	0 1	1 0 1	0 0
23	0 1	0 1 0	1 0	56	0 1	1 0 0	1 0
24	1 0	0 0 1	1 0	57	0 0	2 0 0	1 0
25	0 1	0 0 1	1 0	58	0 1	1 0 0	0 1
26	0 0	1 0 1	1 0	59	0 0	2 0 0	0 1
27	0 1	0 0 1	0 1	60	1 0	2 0 0	1 0
28	0 0	1 0 1	0 1	61	1 0	1 1 0	1 0
29	0 0	0 1 1	0 1	62	0 1	1 1 0	1 0
30	0 0	1 0 0	1 1	63	0 1	1 1 0	0 1
31	0 0	0 1 0	1 1	64	0 1	1 0 1	0 1
32	0 0	0 0 1	1 1	65	1 1	0 1 0	0 0

[9]This reader should notice that Definition 3 further implies that, for any correct linear DAP Δ, $\hat{S}_a(\Delta) \subseteq \hat{S}_{rs}$.

[10]In order to avoid an "over-loading" of the employed notation, in the following we shall use S_a instead of $S_a(\Delta)$, assuming that this set is defined by an appropriately selected DAP Δ.

[11]The application of the employed DAP Δ ensures that every state $\mathbf{s} \in S_a$ will possess at least one enabled event that is also admissible by the applied DAP Δ.

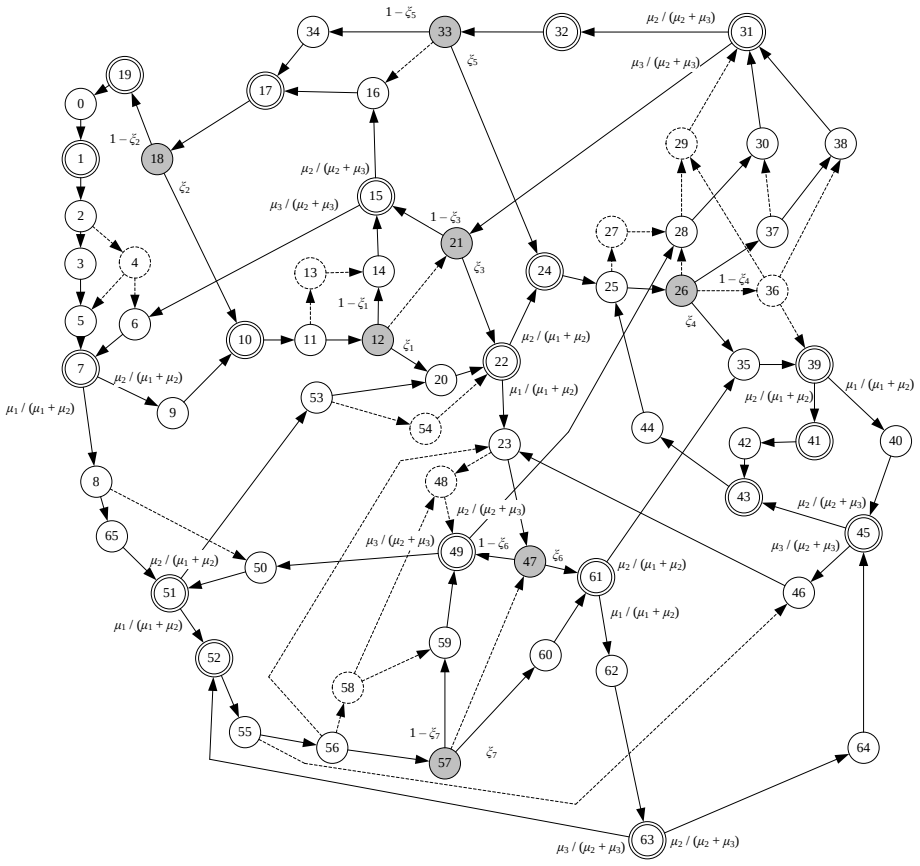

Fig. 3: The reachable and safe state space S_{rs} for the CRL of Figure 2, and some further structure that defines the MDP characterizing the corresponding throughput-maximization problem.

processing stage J_1, and a part p_2 executing processing stage J_2. The earlier completion of each of these two parts w.r.t. the other one leads respectively to states #8 and #9. Furthermore, since processing times for these two parts are exponentially distributed with respective instantaneous rates μ_1 and μ_2, the respective probabilities for each of these two transitions are those annotated in the figure. Finally, the expected sojourn time for state #7 is $1/(\mu_1 + \mu_2) > 0$.

On the other hand, states #8 and #9 are vanishing states, since in each case, the completed part can be advanced to its next processing stage, obtaining, respectively, states #65 and # 10. Furthermore, under the stated operational assumptions for the considered CRL, these part advancements require zero time; hence, the sojourn time for states #8 and #9 is zero. □

As revealed in the previous example, tangible states essentially define an "exponential race" among its enabled events e^d_j, and therefore, it possesses a non-zero sojourn time. On the other hand, since (i) vanishing states enable some controllable event, according to Definition 4, and (ii) any controllable event in the considered CRL model executes in zero time, the sojourn times of these states will be consistently equal to zero.[12]

[12]These remarks also justify the respective names of these two state classes as "tangible" and "vanishing"; the corresponding terminology has been borrowed from [35].

Furthermore, it is important to notice that, while in the case of tangible states the selection of the executed events is resolved endogenously, by the corresponding exponential race, in the case of vanishing states, there is a further need for an extraneous mechanism that will select a particular enabled controllable event for execution. This mechanism must bias the underlying selection in a way that it supports the pursued objective of throughput maximization, and it will be provided by the sought scheduling policy. Next we discuss how this scheduling policy can be obtained, at least in principle, through the formulation and solution of a *Continuous-Time, Average-Reward (CT-AR) MDP* [10] that is defined by means of the various structural elements that have been introduced in this section.

Formulating the considered scheduling problem as an MDP: According to the general MDP theory [10], in order to obtain a complete MDP formulation of the considered scheduling problem in the context of the timed CRL dynamics that were presented in the previous part of this subsection, we need to define: (i) the decision states of this MDP; (ii) the set of actions that are available at each decision state; (iii) the transitional dynamics that are incurred by the execution of a particular action at a decision state; (iv) the immediate rewards that result from these executions; and (v) the function of these rewards that formalizes the problem objective. Next, we provide a detailed characterization of all these elements; our

discussion relies heavily on the various concepts and insights that were provided in the earlier parts of this section.

When it comes to the *"decision states"* of the considered MDP, it should be evident from the discussion that was provided in the previous part of this subsection, that these states are the vanishing states that result from the occurrence of an event e_j^d at any of the admissible tangible states $\mathbf{s} \in S_a^T$. The following definition formalizes this remark.

Definition 5: Let $X \subseteq S_a$ denote the set of states that result from the execution of an event e_j^d at some state $\mathbf{s} \in S_a^T$. Then, set X constitutes the set of *the decision states* of the considered MDP. $\square$

Next, we define the set of the available *"actions"* (or *"decisions"*) at any given state $\mathbf{s}' \in X$. From a more conceptual standpoint, each of these decisions will take the considered MDP to a new admissible tangible state $\mathbf{s}'' \in S_a^T$, in zero time, and then, the process will wait for the next completion event, e_j^d, at that state. It is also important to notice that in order to reach a next admissible tangible state $\mathbf{s}'' \in S_a^T$ from the current decision state $\mathbf{s}'$, the underlying process might have to pass through a cascade of vanishing states that are reached through a sequence of controllable events, σ; the reader is referred to Figure 3 and the accompanying Table I for some concrete examples of this last statement. Furthermore, the above remarks motivate the following definition:

Definition 6: For any decision state $\mathbf{s}' \in X$, define the corresponding *"tangible reach"* of $\mathbf{s}'$, $\mathcal{TR}(\mathbf{s}')$, as the set of the admissible tangible states $\mathbf{s}''$ that are reachable from state $\mathbf{s}'$ through an event sequence $\sigma \in E^*$ that contains only controllable events. Then, the set of *actions*, $\mathcal{A}(\mathbf{s}')$, of the considered MDP at decision state $\mathbf{s}'$ is defined as $\mathcal{A}(\mathbf{s}') \equiv \mathcal{TR}(\mathbf{s}')$. $\square$

For any state $\mathbf{s}'' \in \mathcal{TR}(\mathbf{s}')$, the corresponding action can be materialized through the execution of any controllable-event sequence σ leading from state $\mathbf{s}'$ to state $\mathbf{s}''$.

We also notice, for completeness, that the set of vanishing states $\mathbf{s}'''$ which are reachable from state $\mathbf{s}'$ through the aforementioned event sequences σ that lead to some state $\mathbf{s}'' \in \mathcal{TR}(\mathbf{s})$, is characterized as the *"vanishing reach"* of state $\mathbf{s}'$. This set of states is denoted by $\mathcal{VR}(\mathbf{s}')$, and it can be empty for some states $\mathbf{s}'$.

The *"transitional dynamics"* for the considered CT-MDP model that result from the execution of an action $a \equiv \mathbf{s}'' \in \mathcal{TR}(\mathbf{s}')$ at some decision state $\mathbf{s}'$, are determined by the exponential race that takes place in the tangible state $\mathbf{s}''$.

On the other hand, since our stated objective is the maximization of the long-term throughput of the line, the *expected immediate reward*, $r(\mathbf{s}', a)$, from executing action a at state $\mathbf{s}'$ is defined as follows:

Definition 7: For the considered MDP, the *expected immediate reward* from executing action a at state $\mathbf{s}'$ is denoted by $r(\mathbf{s}', a)$, and it is equal to the probability that the next decision state will be defined by the occurrence of event e_M^d that corresponds to the completion of the last processing stage by a running part and the unloading of this part from the line. Hence, letting $\mathbf{s}'' \in \mathcal{TR}(\mathbf{s}')$ denote the tangible state that corresponds to action a, and $\mathcal{E}(\mathbf{s}'')$ denote the set of the events e_j^d enabled in $\mathbf{s}''$, the expected immediate reward $r(\mathbf{s}', a)$ will be equal to $\mu_M / \sum_{e_j^d \in \mathcal{E}(\mathbf{s}'')} \mu_j$ if $e_M^d \in \mathcal{E}(\mathbf{s}'')$, and zero otherwise.

Finally, in view of the above definitions of the decision states and actions of the considered MDP, the induced transitional dynamics, and the expected immediate rewards, the problem of maximizing the throughput of the considered CRLs is reduced to the problem of maximizing the (long-term) average reward of this MDP.[13] The communicating structure of the admissible state space S_a for the underlying stochastic process that was established in the earlier parts of this section, further implies that this CT-MDP formulation is well defined, and it will have an optimal solution that takes the form of a deterministic, stationary policy [10]. Hence, letting Π denote the set of deterministic stationary policies for the considered MDP, an optimal schedule for the considered CRL is represented by a policy $\pi^* \in \Pi$ that, at each state $\mathbf{s}' \in X$, will select a single action $a \in \mathcal{TR}(\mathbf{s}')$ so that

$$\pi^* = \arg\max_{\pi \in \Pi} \lim_{N \to \infty} \frac{1}{E[t_N]} E\left[\sum_{i=1}^{N} r(\mathbf{s}_i', a_i) \;\middle|\; \mathbf{s}_0, \pi \right]$$

In the above equation, t_N denotes the time of the N-th state transition of the underlying stochastic process. Furthermore, some simplification of this CT-MDP formulation, and some methodology for its solution through uniformization [5], are presented in Appendix A of [15]. But, as remarked in the introductory section, in most practical cases the solution of this CT-MDP model will be intractable due to the very large size of the involved state spaces. Hence, there is a remaining need for the computation of suboptimal scheduling policies that will trade off some of the performance of the underlying system for computational tractability. Such a methodology is developed in the rest of this work. But before we delve into these developments, we conclude this section by discussing the MDP formulation and its optimal solution for the example CRL of Figure 2.

Example: As already discussed in the previous parts of this section, the STD of Figure 3 highlights the classification of the depicted states into tangible and vanishing, and it also reports the transitional dynamics that are defined by the exponential races that take place at each tangible state. An additional development in Figure 3 is a proposed "thinning" of the presented STD through the elimination of the vanishing states and their interconnected transitions that are depicted in dashed lines. This simplification is justified by the facts that (i) the timed performance of the considered CRL is determined by the sojourn times that are spent by this line at the tangible states only, and (ii) the proposed "thinning" does not alter the reachability among the various tangible states, and also among the decision states $\mathbf{s} \in X$ that result from the execution of an

[13] The specification of the set of actions at each decision state $\mathbf{s}'$ of this CT-MDP through the corresponding tangible reach $\mathcal{TR}(\mathbf{s}')$ implies a *non-idling* scheduling policy for the underlying CRL; i.e., under such a policy, no server that could be engaged in the processing of some available part will remain idle. Due to the blocking experienced in the operation of the considered CRLs, such a non-idling scheme might be suboptimal [36], [30]. We have opted to confine the presented developments within the class of the non-idling scheduling policies, in an effort to attain some simplicity for the presentation of the main concepts and ideas involved. But it is possible to extend the presented methodology to *deliberately idling* schemes, by introducing further actions at the states $\mathbf{s}'$ that correspond to decision epochs; these actions will correspond to controllable-event sequences σ leading to some state $\mathbf{s}''$ in the vanishing reach $\mathcal{VR}(\mathbf{s}')$ of the considered state $\mathbf{s}'$, that contains some enabled events e_j^d.

e_j^d–type event.[14]

The vanishing states that involve choice in the remaining STD structure are the seven states colored in grey in Figure 3. More specifically, in the remaining STD, each of these seven states possesses two enabled transitions, and any selection between these two transitions can be represented by setting the corresponding variable $\xi_k,\ k = 1, \ldots, 7$, either to the value of 0 or 1. In the context of the corresponding MDP terminology, any possible pricing of the variables ξ_k depicted in Figure 3 defines a complete deterministic stationary policy for the considered MDP.[15]

On the other hand, to fully specify this MPD, we must also specify the immediate-reward function. Under the previously introduced notation, this function is fully defined by associating with every pair $(\mathbf{s}', \mathbf{s}'') \in X \times \mathcal{TR}(\mathbf{s}')$, an immediate reward equal to the occurrence probability of event $e_3^d (\equiv e^u)$ in state $\mathbf{s}''$.

For a better understanding of the semantics of the MDP that was defined in the previous paragraphs, we also notice that, in the operational context of the CRL depicted in Figure 2, each of the variables ξ_k that define the various deterministic stationary policies for this MDP, essentially models the choice between (i) allocating the server of workstation WS_1 to a part that will execute processing stage J_3 (by setting $\xi_k = 0$), and (ii) allocating this server to a newly loaded part for the execution of its first processing stage (by setting $\xi_k = 1$).

Finally, according to the computational results that are reported in [9], in the case where $\tau_j = 1.0,\ \forall j$, the optimal policy for the aforementioned MDP is uniquely defined by $\xi_k = 1,\ \forall k$, a result that can be interpreted as a "First-Buffer-First-Serve" policy for the considered operational context.

III. The proposed scheduling method

This section presents the heuristic scheduling method for the considered CRL model that is proposed in this work. In the subsequent developments, the considered CRL is assumed to be controlled by a correct *linear* DAP Δ obtained, for instance, through the corresponding methodology that is presented in [6]. Hence, in the context of the RAS real-time control framework of Figure 1, the methodology that is developed in this section supports essentially the function of the performance-oriented controller / scheduler in that figure.

From an organizational standpoint, the section consists of three major subsections, with the first subsection introducing the proposed method itself. The second subsection demonstrates the application and the efficacy of the method through the example CRL of Figure 2. And, finally, the last subsection provides some complexity analysis of the proposed method, highlighting the primary factors that determine this complexity, and establishing, thus, the tractability of the method. Further empirical assessment of the tractability of the method and of the quality of the scheduling policies that are derived by it, are provided in the next section.

A. The proposed scheduling method

The basic structure and rationale of the proposed method: As discussed in the introductory section, the considered scheduling method effects its decisions at each decision state reached by the underlying CRL by (i) first formulating and solving an LP that is known as the corresponding "(fluid) LP relaxation", and (ii) subsequently utilizing the information that is contained in the optimal solution of this LP in order to select an optimized action at that decision point.

The employed LP relaxation can be perceived as a simplification of the underlying scheduling problem that is obtained by treating the material processed through the considered CRL as a *continuous flow* that is constrained by the processing capacities of the line servers. Additional constraints restrict the spatial distribution of this flow across the line workstations and its time-based and evolution so that it observes (i) the buffering capacities of these workstations, and (ii) the additional bounds that are imposed by the applied DAP. In this new operational setting, the considered LP seeks to maximize the line output over a predetermined time horizon T, further assuming that (a) the line workstations are initialized with the fluid levels corresponding to the current decision state $\mathbf{s}$, and (b) there exists an "infinite backlog" of fluid that can be fed into the line.

As demonstrated in the following, assuming that the employed time horizon T is sufficiently long, any optimal solution of the considered LP will seek to drive the line to a "steady-state" operational regime that maximizes its output flow rate, while minimizing the losses that will be experienced during the transient phase. This realization further suggests that the scheduler of Figure 1 can utilize the information about the server allocation that is encoded in the very first part of any optimal solution of the considered LP formulation, as guidance for resolving the action selection at the decision state $\mathbf{s}$ under consideration. The detailed logic for translating the obtained solution of the LP relaxation to an action-selection scheme is the second major component of this method.

As mentioned in the introductory section, LP relaxations similar to that outlined in the previous paragraphs have been employed in the past for the scheduling of uncapacitated re-entrant lines and other more general multi-class queueing networks [17], [18], [20]. On the other hand, the particular implementation of this general method in the CRL (or the more general RAS) context that is presented in this work, is differentiated from the previous instantiations of the method in the aforementioned references by (i) the integration of this scheduling methodology with the necessary LES policies in the real-time control framework of Figure 1, and (ii) the ensuing need for further modification of the method in order to accommodate effectively the spatio-temporal constraints that are imposed by the workstation buffering capacities and the employed DAP.

[14] A complete analysis that formalizes this "thinning" process, provides a more rigorous justification for it, and supports it with computationally efficient algorithms, is presented in [14].

[15] In more accurate terms, the suggested pricing of the variables ξ_k defines a deterministic stationary policy of the considered MDP because of the following two additional facts: (I) For any vanishing state $\mathbf{s}' \in X$, that results from the execution of an e_j^d–type event at some tangible state $\mathbf{s}$, the aforementioned pricing of the variables ξ_k defines completely the next tangible state $\mathbf{s}''$ to be reached from state $\mathbf{s}'$. (II) In addition, for any pair of states $\mathbf{s}'_1$ and $\mathbf{s}'_2$ in X, the pricing of the variables ξ_k does not introduce any "coupling" in the resulting transitional dynamics from these two states.

We also notice that allowing the variables ξ_k to take values in the interval $[0, 1]$ would result in a randomized stationary scheduling policy. But in the considered problem setting, enabling such a randomization will not lead to any performance enhancement for the underlying CRL [10], [31].

In the rest of this subsection we provide all the necessary details fo a complete implementation of the considered scheduling method in the CRL context. This discussion will also show that the sought integration to the employed LP relaxation of the spatio-temporal restrictions that are imposed by the workstation buffering capacities and the employed DAP, in a way that captures effectively the impact of these restrictions on the underlying operation of the line, requires some special care. Hence, one first differentiation of our implementation of the method compared to its past implementations in [17], [18], [20] that results from the aforementioned consideration, is the modeling of the fluidized operation of the line, and the definition of the corresponding LP formulation, in a discrete-time setting. We turn to this issue next.

Time discretization: In an effort to capture more effectively the impact of the blocking effects that are caused by the finite buffers and the imposed DAP, our LP relaxation is formulated in discrete and not in continuous time. More specifically, assuming that the processing times τ_j for the different processing stages $J_j,\ j = 1, \ldots, M$, are rationally valued, we set the discretizing time interval Δt equal to the greatest common divisor (GCD) of τ_j. In this way, the mean processing time, τ_j, of any processing stage J_j, corresponds to an integral multiple of Δt, which will be denoted by $\hat{\tau}_j$. In the following discussion, we also scale time by further assuming that $\Delta t = 1.00$, and thus, $\hat{\tau}_j$ also denotes the mean processing time of processing stage J_j in this new time scale. The significance of this discretization in the context of the pursued modeling will be revealed in the detailed discussion of the employed LP formulation, which is the topic to be considered next.

The employed LP relaxation: We start the detailed presentation of the employed LP relaxation, by introducing first some supporting notation. Subsequently, we introduce the decision variables, the constraints, and the objective function, in this order.

Supporting notation:

- $\mathcal{J}_l,\ l = 1, ..., L$: The set of all processing stages executed on workstation WS_l; i.e., $\mathcal{J}_l = \{j : W(J_j) = l, j = 1, \ldots, M\}$.
- T: The total time horizon over which we are maximizing the line throughput; as explained in the opening part of this section, T is expressed in terms of the discretizing time interval Δt.
- $\mathbf{s}^{init}$: The CRL vanishing state that corresponds to the current decision state.
- $\mathbf{v}$: A $2M$-dim vector with its components $\mathbf{v}_{1+2j},\ j = 0, \ldots, M-1$, representing the volume of "fluid" waiting for the execution of processing stage J_{j+1} at the corresponding workstation $W(J_{j+1})$, and the components $\mathbf{v}_{2+2j},\ j = 0, \ldots, M-1$, representing the volume of "fluid" that has completed the execution of processing stage J_{j+1} but it is still located at the corresponding workstation $W(J_{j+1})$. We shall refer to the components of vector $\mathbf{v}$ as the corresponding "fluid buffers".
- $\mathbf{v}^{init}$: The initial value for the "buffer fluid" vector $\mathbf{v}$ as defined by the state vector $\mathbf{s}^{init}$. Due to the presumed exponential nature for the distribution of the various processing times, components $\mathbf{v}_{1+2j},\ j = 0, \ldots, M-1$, will aggregate all the parts that either wait for the initiation of the execution of the corresponding processing stage J_{j+1} or have already initiated the execution of this processing stage.
- f: A fictitious "fluid feeder" at the beginning of the line representing an "infinite backlog".
- d: A fictitious "fluid buffer" at the end of the CRL, of unlimited capacity, that collects all the "fluid" that is output by this line over the considered time horizon T.

Decision Variables:

- $x_{j,t}, j = 1, \ldots, 2M, t = 1, \ldots, T$: The "fluid" volume in "fluid buffer" $\mathbf{v}_j$ at the end of period t.
- $u_{j,t}, j = 1, \ldots, 2M+1, t = 1, \ldots, T$: The amount of the "fluid" that is added, during period t, to the "fluid" buffer $\mathbf{v}_j$ or, in the case of $j = 2M+1$, to the "output fluid buffer" d. More specifically:
  - $u_{1,t}$ represents the amount of "fluid" that is added to the "fluid buffer" $\mathbf{v}_1$ at period t. This "fluid" is drawn from the external "fluid feeder" f, during the same period, and its addition to the "fluid buffer" $\mathbf{v}_1$ is equivalent to the action of loading new material to the CRL.
  - $u_{2+2i,t}, i = 0, ..., M-1$, represent the amount of "fluid" that is added to the corresponding "fluid buffer" $\mathbf{v}_{2+2i}$ at period t. This "fluid" corresponds to material completing the processing of processing stage J_{i+1}, and it was drawn from "fluid buffer" $\mathbf{v}_{1+2i}$ at period $t - \hat{\tau}_{i+1} + 1$.
  - $u_{1+2i,t},\ i = 1, \ldots, M-1$, represent the amount of "fluid" that is added to the corresponding "fluid buffer" $\mathbf{v}_{1+2i}$ at period t. This "fluid" corresponds to material transferred to this "fluid buffer" from "fluid buffer" $\mathbf{v}_{2i-1}$ during this period.
  - $u_{2M+1,t}$ represents the amount of "fluid" that is transferred from the "fluid buffer" $\mathbf{v}_{2M}$ to the "output fluid buffer" d during period t.

Constraints:

1) The first set of constraints expresses the limited processing capacity at each workstation; namely, the server at each workstation cannot process more than a unit amount of work during a single time unit.

$$\sum_{j \in S_l} \sum_{q=t}^{\min\{t+\hat{\tau}_j-1, T\}} u_{2j,q} \ \leq\ 1,\ l = 1, ..., L,\ t = 1, ..., T$$

2) The second set of constraints expresses the material flow conservation; these constraints break down into the following two parts:

   a) Material flow conservation constraints for period $t = 1$:
   $x_{1+2i,1} = \mathbf{v}^{init}_{1+2i} + u_{1+2i,1} - u_{2+2i,\hat{\tau}_{i+1}} \mathbf{1}_{\{\hat{\tau}_{i+1} \leq T\}},\ i = 0 \ldots, M-1$
   $x_{2i,1} = \mathbf{v}^{init}_{2i} + u_{2i,1} - u_{1+2i,1},\ i = 1 \ldots, M$

   b) Material flow conservation constraints for periods $t = 2, \ldots, T$:
   $x_{1+2i,t} = x_{1+2i,t-1} + u_{1+2i,t} - u_{2+2i,t+\hat{\tau}_{i+1}-1} \mathbf{1}_{\{t+\hat{\tau}_{i+1}-1 \leq T\}},\ i = 0 \ldots, M-1$
   $x_{2i,t} = x_{2i,t-1} + u_{2i,t} - u_{1+2i,t},\ i = 1 \ldots, M$

3) This set of constraints expresses the fact that a server cannot work on an empty buffer, while also acknowledging the availability of the "infinite backlog" that provides the input material for processing stage J_1; similar to the second set of constraints, we express these constraints separately for period 1 and for the remaining periods:
   a) For period $t = 1$:
   $\mathbf{v}^{init}_{1+2i} + \mathbf{v}^{init}_{2i} - u_{2+2i,\hat{\tau}_{i+1}} \mathbf{1}_{\{\hat{\tau}_{i+1} \leq T\}} \geq 0$, $i = 1 \ldots, M-1$
   b) For periods $t = 2, \ldots, T$:
   $x_{1+2i,t-1} - u_{2+2i,t+\hat{\tau}_{i+1}-1} \mathbf{1}_{\{t+\hat{\tau}_{i+1}-1 \leq T\}} \geq 0$, $i = 1 \ldots, M-1$
4) These constraints express the finite buffering capacity of the line workstations.
   $\sum_{j \in S_l} x_{2j-1,t} + x_{2j,t} + \sum_{q=t+1}^{\min\{t+\hat{\tau}_j-1, T\}} u_{2j,q} \leq B_l$, $l = 1, ..., L, \quad t = 1, ..., T$
5) These constraints account for the imposed deadlock avoidance policy Δ. The presumed linear structure of the applied DAP Δ implies that the state-admissibility condition of the policy can be expressed as a set of K inequalities having the form
$$A \cdot \hat{\mathbf{s}} \leq \mathbf{b} \tag{5}$$
   where: (i) $\hat{\mathbf{s}}$ is the condensed state of the considered CRL, (ii) A is a $K \times M$ matrix, and (iii) $\mathbf{b}$ is a K-dim positive vector. The constraints of Eq. 5 can be introduced in the considered LP relaxation by substituting each component $\hat{\mathbf{s}}_j,\ j = 1, \ldots, M$, of the state vector $\hat{\mathbf{s}}$ by the quantity
$$x_{2j-1,t} + x_{2j,t} + \sum_{q=t+1}^{\min\{t+\hat{\tau}_j-1, T\}} u_{2j,q}$$
6) We also want to prevent activity that will not contribute to the total output volume by the end of the time horizon T. For this, we enforce the condition that the total outflow from the network equals the total inflow to it plus the initial "fluid buffer" contents as defined by the vector $\mathbf{v}^{init}$.
$$\sum_{j=1}^{2M} \mathbf{v}^{init}_j + \sum_{t=1}^{T} u_{1,t} - \sum_{t=1}^{T} u_{2M+1,t} = 0$$
7) This constraint recognizes the fact that "fluid buffer" contents cannot be negative.
$$x_{j,t} \geq 0, \quad j = 1, ..., 2M, \quad t = 1, ..., T$$
8) Also, the "material flows" $u_{j,t}$ cannot be negative either.
$$u_{j,t} \geq 0, \quad j = 1, ..., 2M+1, \quad t = 1, ..., T$$
9) Finally, the next constraint accounts for the non-preemptive nature of our scheduling policies.[16]
$$u_{2j,\hat{\tau}_j} = 1, \quad j \in \{1, \ldots, M : \mathbf{s}^{init}_{1+3(j-1)} = 1\}$$

Objective Function:

As already stated, we want to maximize the total outflow of the considered CRL over the employed time horizon T, assuming that (i) the line is operated under the relaxed modeling assumptions that are expressed by the constraints of the considered LP, and (ii) its initial "fluid buffer" contents are set to the levels that are defined by the state $\mathbf{s}^{init}$ of the original CRL model. Hence, the objective function takes the form:
$$\max \sum_{t=1}^{T} u_{2M+1,t}$$

The induced scheduling policy: After we have solved the LP relaxation, the next step is to interpret the solution of the linear program to a scheduling policy for the underlying CRL. In particular, we want to use the solution of this LP as a "guide" in the selection of the next tangible state $\mathbf{s}$ among the set of tangible states that is defined by the tangible reach, $\mathcal{TR}(\mathbf{s}^{init})$, of the state $\mathbf{s}^{init}$ that constitutes the current decision point.

To effect this selection, let us denote by $\mathbf{u}^*_1$ the vector that is defined by the obtained optimal values for the variables $u_{1,1}, u_{2,\tau_1}, u_{3,1}, u_{4,\tau_2} \ldots, u_{2M,1}$, and by $\mathbf{v}$ the "fluid buffer" vector that corresponds to any state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})$. Then, the proposed scheduling policy will select the next tangible state, $\tilde{\mathbf{s}}$, through the following rule:
$$\tilde{\mathbf{s}} \in \arg \min_{\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})} \sum_{j=0}^{M-1} |\mathbf{s}_{1+3j} - \mathbf{u}^*_{1,2+2j}| \tag{6}$$

In more natural terms, the criterion of Eq. 6 seeks to select a tangible state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})$ that has a server allocation w.r.t. the various processing stages $J_j,\ j = 1, \ldots, M$, that is most similar to the server allocation that is implied by the vector $\mathbf{u}^*_1$.

Furthermore, a secondary criterion that we have used to break any ties that are generated through the criterion of Eq. 6, is as follows:
$$\tilde{\mathbf{s}} \in \arg \min_{\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})} |\mathbf{v} - \mathbf{v}^{init} - \mathbf{u}^*_1|_1 \tag{7}$$

This new criterion perturbs the initial "buffer fluid" vector $\mathbf{v}^{init}$ by the "flow" vector $\mathbf{u}^*_1$, and eventually selects a tangible state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})$ with a "fluid buffer" vector $\mathbf{v}$ that has the smallest l_1-distance from the aforementioned perturbation $\mathbf{v}^{init} + \mathbf{u}^*_1$; hence, this secondary criterion considers also state similarity in terms of buffer occupancy.

Selecting an appropriate time-horizon length T: One last parameter that needs to be further specified for the complete definition of the CRL scheduling methodology that was presented in the previous parts of this section, is the value of the parameter T to be employed in the LP relaxation, i.e., the time-horizon over which the line output will be maximized. This selection is driven by the realization that the optimal solution of the considered LP will essentially lead the system to an operational regime that provides the maximal possible output of the system as defined by the bottleneck stations of the line and the applied DAP bounds, and it will divert from this operational regime only towards the end of the operational horizon, in an effort to satisfy the termination condition of Constraint #6 above. Furthermore, the numerical experimentation that is reported in the last part of this paper,

[16] In the MDP formulation of Section II-D, the non-preemptive character of the pursued policies is implied by the structure of the underlying state space S and the dynamics that are induced by this structure for that formulation.

TABLE II: Comparing the policy specified for he example CRL of Figure 2 by the methodology that is presented in this work to the optimal policy for this re-entrant line.

s	Vanishing State			Tangible Reach			Opti-mal	Select. Crit. of Eq. 6
	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$		
12	00	100	10	10	010	10	YES	1.0626
				00	010	01	NO	1.4439
18	00	000	10	10	000	10	YES	0.7693
				00	000	01	NO	2.3220
21	00	010	10	10	010	10	YES	0.7285
				00	010	01	NO	1.2715
26	00	101	10	10	011	10	NO	1.7492
				00	010	11	YES	1.2365
33	00	001	10	10	001	10	YES	0.8282
				00	000	11	NO	1.7219
47	00	110	10	00	110	01	YES	0.6529
				10	110	10	NO	1.3471
57	00	200	10	00	110	01	YES	0.6826
				10	110	10	NO	1.5370

has shown that as long as the selected T value is adequately large to let the line reach the aforementioned operational regime, the returned vector $\mathbf{u}_1^*$ that is used in the determination of the induced scheduling policy, will be quite insensitive to the exact T value. So, with these insights and findings, we propose to set $T = (\sum_{i=1}^{L} B_i)(\sum_{j=1}^{M} \hat{\tau_j})$, since this is an upper bound of the time that is necessary to empty the line from the entire workload that is defined by the state $\mathbf{s}^{init}$ under any globally nonidling policy.

B. Example

The application of the "fluid" LP relaxation that was presented in the earlier parts of this section at any of the seven vanishing states $\mathbf{s}_l,\ l \in \{12, 18, 21, 26, 33, 47, 57\}$, that constitute decision points for the CRL of Figure 2 (c.f. also the STD of Figure 3), results in the following LP formulation:

$$\max_{x,u} \quad \sum_{t=1}^{T} u_{7,t}$$

s.t.

$$\begin{aligned}
u_{2,t} + u_{6,t} &\le 1, \quad t = 1, \ldots, T \\
u_{4,t} &\le 1, \quad t = 1, \ldots, T \\
x_{1+2i,1} - u_{1+2i,1} + u_{2+2i,\hat{\tau}_{i+1}} &= \mathbf{v}_{1+2i}^{init}, \quad i = 0, \ldots, 2 \\
x_{2i,1} - u_{2i,1} + u_{1+2i,1} &= \mathbf{v}_{2i}^{init}, \quad i = 1, \ldots, 3 \\
x_{1+2i,t} - x_{1+2i,t-1} - u_{1+2i,t} + u_{2+2i,t+\hat{\tau}_{i+1}-1} &= 0, \\
& \quad i = 0, \ldots, 2, \quad t = 2, \ldots, T \\
x_{2i,t} - x_{2i,t-1} - u_{2i,t} + u_{1+2i,t} &= 0, \\
& \quad i = 1, \ldots, 3, \quad t = 2, \ldots, T \\
u_{2+2i,1} &\le \mathbf{v}_{1+2i}^{init} + \mathbf{v}_{2+2i}^{init}, \quad i = 1, 2 \\
u_{2+2i,t} - x_{1+2i,t-1} &\le 0, \quad i = 1, 2, \quad t = 2, \ldots, T \\
x_{1,t} + x_{2,t} + x_{5,t} + x_{6,t} &\le 2, \quad t = 1, \ldots, T \\
x_{3,t} + x_{4,t} &\le 2, \quad t = 1, \ldots, T \\
x_{1,t} + x_{2,t} + x_{3,t} + x_{4,t} &\le 3, \quad t = 1, \ldots, T \\
\sum_{t=1}^{T} u_{7,t} - \sum_{t=1}^{T} u_{1,t} &= \sum_{j=1}^{6} \mathbf{v}_j^{init} \\
x_{i,t} &\ge 0, \quad i = 1, \ldots, 6, \quad t = 1, \ldots, T \\
u_{i,t} &\ge 0, \quad i = 1, \ldots, 7, \quad t = 1, \ldots, T \\
u_{2j,\hat{\tau}_j} &= 1, \quad j \in \{1, 2, 3 : \mathbf{s}_{1+3(j-1)}^{init} = 1\}
\end{aligned}$$

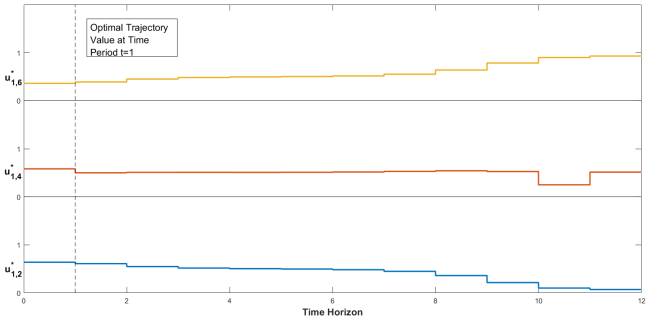

Fig. 4: The optimal server allocation, over the entire time horizon T, that is returned by the solution of the "fluid" relaxation for the example CRL of Figure 2 at the vanishing state $\mathbf{s}_{12}$.

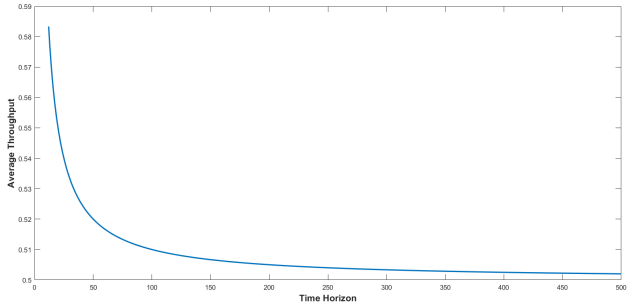

Fig. 5: The average throughput obtained through the solution of the "fluid" relaxation for the example CRL of Figure 2 over different time horizons T; the starting state of the line is the vanishing state $\mathbf{s}_{12}$.

The parameters $\mathbf{s}_{\cdot}^{init}$ and $\mathbf{v}_{\cdot}^{init}$ that appear in the right-hand-side of the above formulation, are determined by the considered vanishing state $\mathbf{s}_l$ according to the defining logic for these parameters that was discussed during their introduction in the earlier parts of this section.

Table II presents the policy that is defined by the solution of the above LP formulation at the seven vanishing states $\mathbf{s}_l,\ l \in \{12, 18, 21, 26, 33, 47, 57\}$, of this example CRL when $\tau_j = 1.0,\ \forall j$. Also, in line with our earlier recommendations, in the corresponding computations the parameter T was set equal to $4 \times 3 = 12$. Each primary row in Table II corresponds to one of the considered vanishing states $\mathbf{s}_l$, and the first two parts of the row provide a complete characterization of state $\mathbf{s}_l$ and its tangible reach, $\mathcal{TR}(\mathbf{s}_l)$. On the other hand, the row entry in the column entitled "Optimal" provides the choice for the next tangible state $\mathbf{s}'_l \in \mathcal{TR}(\mathbf{s}_l)$ specified by the optimal policy that is obtained through the solution of the throughput-maximizing MDP formulation for this CRL of Section II-D. Finally, the last column of Table II provides the values for the "action"-selection criterion of Equation 6 that are obtained from the solution of the corresponding LP relaxations. It can be checked that, for each state $\mathbf{s}_l,\ l \in \{12, 18, 21, 26, 33, 47, 57\}$, the minimum value for this criterion corresponds to the tangible state $\mathbf{s}'_l \in \mathcal{TR}(\mathbf{s}_l)$ that is the

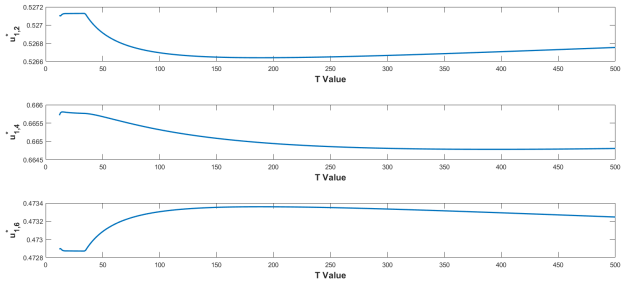

Fig. 6: The values of the vector $\mathbf{u}_1^*$ obtained from the solution of the "fluid" relaxation for the example CRL of Figure 2, over different time horizons T; the starting state of the line is the vanishing state $\mathbf{s}_{12}$.

optimal choice according to column "Optimal". Hence, for this example CRL, our scheduling methodology is able to identify an optimal policy.[17]

Figure 4 depicts the evolution of the variable sequences $u_{j,t}^*, t \in \{0, \ldots, 12\}$, $j = 2, 4, 6$, that constitute part of the optimal solution of the "fluid" relaxation for the example CRL of Figure 2 when this line is started at the vanishing state $\mathbf{s}_{12}$. In the operational context of the considered CRL, these three sequences essentially represent the server allocation that is implied by the optimal solution of the "fluid" relaxation. The three plots of Figure 4 exhibit clearly that the optimal solution of this relaxation drives the line to a workflow configuration where all servers maintain a constant allocation for the most part of the corresponding time horizon, except for some starting and ending phases where the solution must satisfy the specified boundary conditions. Similar behavior is exhibited for the remaining vanishing states $\mathbf{s}_l$ that constitute decision points for this line. Also, Figure 5 corroborates to the above remarks, by showing that as the value of the time horizon T is increased to ever higher values, the average line throughput, under the optimal solutions of the corresponding LP formulations, converges to the value of 0.5, which is the production rate of the "bottleneck" workstation WS_1. Finally, Figure 6 exhibits the server allocation for period $t = 1$ that is specified by the optimal solution of the "fluid" relaxation for the considered CRL, using a set of values for T that ranges from 12 to 500 periods. It is clearly seen in the provided plots that the corresponding u_1^* vectors are practically insensitive to this variation of the parameter T; this fact further implies that the scheduling policy specified by the criterion of Equation 6 will be insensitive to this variation of T, as well.

C. Complexity considerations

In this subsection we provide some remarks that establish the tractability of the proposed scheduling method, and also reveal the primary factors that determine its computational efficiency. Also, the last part of the section outlines some additional possibilities that can be employed during the method implementation, in case that there is a need to alleviate the computational overhead that is incurred by the involved computation.

We start the overall discussion by noticing that at core of the proposed method is the solution, at each decision epoch, of the LP relaxation that was developed in Section III-A. This LP formulation involves

- $(4M + 1)T = (4M + 1)(\sum_{i=1}^{L} B_i)(\sum_{j=1}^{M} \hat{\tau}_j)$ variables, and
- $(2L + 2M - 1 + K)T + x + 1$ technological constraints.

To help the reader parse the above expressions, we also remind her that, according to the adopted notation, M stands for the number of processing stages, L stands for the number of the line workstations, T is the length of the employed time horizon in the discretizing time unit Δt, B_l, $l = 1, \ldots, L$, is the buffer size at the workstation W_l, and $\hat{\tau}_j$, $j = 1, \ldots, M$, is the mean processing time of processing stage J_j under the time discretization and normalization that were introduced in Section III-A. Also, x implies the number of active servers at the current decision state $\mathbf{s}^{init}$.

Then, when we also consider the computational capabilities of the current commercial LP solvers, it is clear from the above expressions, that the generated LPs will be effectively solvable by these LP solvers for a very broad spectrum of CRL configurations. This is especially true when we realize that the values $\hat{\tau}_j$ that appear in the computation of the employed time horizon T, are normalized w.r.t. the gcd of the actual mean processing times τ_j, $j = 1, \ldots, M$; hence, as long as the original mean processing times do not have a very large spread, then the $\hat{\tau}_j$ values that are employed in the formulation can be pretty small.[18] The above remarks are corroborated by a series of numerical experiments that are reported in Section IV and reveal that the necessary solution times for the proposed LP relaxation are in the order of a few seconds even for some pretty sizeable CRL configurations.

Yet, an additional concern arises from the fact that the considered LP relaxation must be solved "on-line" and in a repetitive fashion, i.e., at each decision point that is encountered during the real-time operation of the underlying CRL. The repetitive solution of this LP defines a computational overhead that can be significant, especially in the case that the processing times involved are pretty small, and therefore, the solution times of the formulated LPs are comparable to these processing times. If this happens to be the case, then the resulting computational overhead can be controlled through the "hashing" of the LP solutions in combination with some approximating / interpolating schemes similar to those that are outlined in [20] for the implementation of the LP-relaxation-based scheduling method that is discussed in that work; we refer the reader to [20] for some discussion on these schemes and the corresponding implementational details.

[17] We emphasize, however, that the considered methodology does not precompute the applied scheduling policy in the form that is communicated in Table II. At each decision state, the action that is selected by this policy is determined in real-time through (i) the formulation and solution of the corresponding LP relaxation, and (ii) the post-processing of the obtained optimal solution for this relaxation through the selection logic of Equations 6 and 7.

[18] As a more vivid example of this statement, in the case of CRLs where the original mean processing times τ_j, $j = 1, \ldots, M$, are equal to some constant value C, all the corresponding $\hat{\tau}_j$ will be equal to 1.0, irrespective of what is the actual value of C.

IV. Some numerical experiments

In this section we report two series of experiments. The first set of experiments intends to further demonstrate and assess the ability of the proposed scheduling method to return scheduling policies that are (a) of comparable quality to the corresponding optimal scheduling policies, and (b) very competitive w.r.t. some other heuristic scheduling policies that are adapted from the corresponding literature. On the other hand, the second set demonstrates and assesses the computational tractability of the presented method. We organize the relevant material into two separate subsections.

A. Demonstrating and assessing the quality of the obtained schedules

In this part of the presented experiments, we used the 20 CRL configurations that are listed in Table III in order to assess the performance of the scheduling methodology that has been developed in this work against (i) the optimized performance that can be obtained (at least, in principle) through the solution of the corresponding MDP formulation of Section II-D, and also (ii) the performance of some heuristic scheduling policies for these lines, that have been adapted from the relevant literature on the throughput maximization of uncapacitated re-entrant lines [28], [29], [37]. More specifically, for each of the CRL configurations 1 to 16 in Table III, we generated 30 problem instances by varying randomly the processing rates for the corresponding processing stages over the interval [1-10]. On the other hand, for each of the CRL configurations 17 to 20 of Table III, we generated only 5 problem instances, with similar ranges for the random processing rates of their processing stages, since the state spaces for these configurations are very large, and therefore, the computation of these state spaces and the performance evaluation of the corresponding scheduling policies took a very long time. Furthermore, in the employed "fluid" relaxations, we set the length of the employed time horizon $T = 20 \sum_{i=1}^{M} \hat{\tau}_j$.

The heuristic scheduling policies that are considered in this experiment, are described as follows:[19]

- **"Fluid"-Relaxation(-based) Policy – FR:** This is the policy defined by the scheduling method that is developed in this work.
- **First-Buffer-First-Serve Policy– FBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that has the line working at the earliest possible processing stage of the line. If there are many such tangible states in $\mathcal{TR}(\mathbf{s})$, the selected state is the one that incurs the largest number of part advancements from their current processing stage to the next one.
- **Last-Buffer-First-Serve Policy – LBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that has the line working at the latest possible processing stage of the line. If there are many such tangible states in $\mathcal{TR}(\mathbf{s})$, the selected state is the one that incurs the largest number of part advancements from their current processing stage to the next one.
- **Shortest-Processing-Time-FBFS – SPT-FBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that leads to the processing of one of the parts with the smallest expected processing time among the parts that can receive processing in the next decision epoch. In case of many such tangible states in $\mathcal{TR}(\mathbf{s})$, the final state is selected according to the FBFS logic that was defined in the first item above.
- **Shortest-Processing-Time-LBFS – SPT-LBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that leads to the processing of one of the parts with the smallest expected processing time among the parts that can receive processing in the next decision epoch. In case of many such tangible states in $\mathcal{TR}(\mathbf{s})$, the final state is selected according to the LBFS logic that was defined in the second item above.
- **Maximum-Pressure Policy – MP:** For any state $\mathbf{s} \in S$ and any processing stage $J_j,\ j = 1, \ldots, M$, we define the "pressure" associated with processing stage J_j at state $\mathbf{s}$ as

$$\mathcal{P}(\mathbf{s}, J_j) \equiv \mu_j[(\mathbf{s}_{3(j-1)} + \mathbf{s}_{3(j-1)-1})I_{\{j>1\}} + \mathbf{s}_{1+3(j-1)} - (\mathbf{s}_{2+3(j-1)} + \mathbf{s}_{3j} + \mathbf{s}_{1+3j})I_{\{j<M\}}]$$

  Also, we define the "(total) pressure" associated with state $\mathbf{s}$ by

$$\mathcal{P}(\mathbf{s}) \equiv \sum_{j=1}^{M} \mathbf{s}_{1+3(j-1)} \mathcal{P}(\mathbf{s}, J_j)$$

  i.e., $\mathcal{P}(\mathbf{s})$ is the total pressure across all processing stages that receive processing at state $\mathbf{s}$.
  Then, at any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, and for any tangible state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$, this policy selects a state $\mathbf{s}'$ that has the highest total pressure among the states in $\mathcal{TR}(\mathbf{s})$.

The performance of each of these heuristic policies for each CRL instantiation that was generated in the considered experiment, was evaluated by solving the LP that is obtained from the corresponding Bellman equation [10] by fixing the selected actions at each decision state $\mathbf{s} \in X$ to the actions that are specified by this policy. In this way, we can characterize the level of sub-optimality for each of these policies, for any given CRL instantiation, through a "percentage (%-) error" that is defined by:

$$\%\text{-error} = \frac{\text{optimal throughput - policy throughput}}{\text{optimal throughput}} \times 100$$

Table IV reports the average, minimum and maximum %-errors that were observed during the application of the

[19] As already mentioned in the opening paragraph of this section, the heuristic policies that have been used as "benchmarks" for the presented experiment, constitute adaptations to the CRL operational setting of some simple policies that have been shown to be throughput-optimal for the uncapacitated re-entrant lines. The performed adaptation seeks to fit the procedural logic that defines the original policies to the operational setting that is considered in this paper. But, of course, these modifications do not extend the original optimality analysis for these policies to this new setting. On the other hand, to the best of our knowledge, there are no other heuristic scheduling policies that are known to be (near-)optimal for the considered operational setting and could have defined a more appropriate "benchmark" for the results that are presented in this paper.

TABLE III: The CRL configurations considered in the numerical experiment of Section IV-A (borrowed from [15]).

Configuration	Number Of Workstations	Number of Job Stages (JS) and Job Routes	Buffer Capacities
Conf 1	2	3JS ($W_1 \rightarrow W_2 \rightarrow W_1$)	$(B_1, B_2) = (1, 2)$
Conf 2			$(B_1, B_2) = (2, 2)$
Conf 3			$(B_1, B_2) = (3, 2)$
Conf 4			$(B_1, B_2) = (4, 4)$
Conf 5			$(B_1, B_2) = (9, 9)$
Conf 6	3	4JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1$)	$(B_1, B_2, B_3) = (1, 2, 2)$
Conf 7			$(B_1, B_2, B_3) = (3, 2, 2)$
Conf 8			$(B_1, B_2, B_3) = (4, 3, 2)$
Conf 9			$(B_1, B_2, B_3) = (5, 5, 6)$
Conf 10	4	7JS($W_1 \rightarrow W_2 \rightarrow W_4 \rightarrow W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1$)	$(B_1, B_2, B_3, B_4) = (3, 2, 1, 2)$
Conf 11	3	5JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1 \rightarrow W_2$)	$(B_1, B_2, B_3) = (3, 4, 3)$
Conf 12	3	5JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_2 \rightarrow W_3$)	$(B_1, B_2, B_3) = (3, 3, 3)$
Conf 13	3	5JS($W_1 \rightarrow W_2 \rightarrow W_1 \rightarrow W_3 \rightarrow W_2$)	$(B_1, B_2, B_3) = (3, 4, 1)$
Conf 14			$(B_1, B_2, B_3) = (2, 2, 2)$
Conf 15	3	6JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1 \rightarrow W_2 \rightarrow W_3$)	$(B_1, B_2, B_3) = (2, 3, 2)$
Conf 16			$(B_1, B_2, B_3) = (2, 2, 2)$
Conf 17	4	7JS($W_1 \rightarrow W_2 \rightarrow W_4 \rightarrow W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1$)	$B_i = 3, i = 1, \ldots, 4$
Conf 18	5	7JS($W_1 \rightarrow W_2 \rightarrow W_1 \rightarrow W_3 \rightarrow W_4 \rightarrow W_5 \rightarrow W_4$)	$B_1 = B_2 = B_3 = 2$ $B_4 = B_5 = 3$
Conf 19	4	8JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_2 \rightarrow W_3 \rightarrow W_4 \rightarrow W_3 \rightarrow W_4$)	$B_i = 3, i = 1, \ldots, 5$
Conf 20	5	8JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_2 \rightarrow W_3 \rightarrow W_4 \rightarrow W_5 \rightarrow W_3$)	$B_i = 3, i = 1, \ldots, 5$

considered scheduling policies on the generated instances from the 20 CRL configurations of Table III. It can be seen that the FR policy results in pretty small %-errors. This policy also outperforms the other heuristic policies in terms of, both, the average and the maximal values of these errors, a result that suggests an ability of this policy to obtain better performance than the other policies in a consistent manner. This assessment was further substantiated by performing a paired t-test [38] and a paired Wilcoxon test [39] on the %-error values that were obtained in the considered experiment. The p-values that were obtained through these two tests, assessing the dominance of the FR policy over each of the remaining heuristic policies, are reported in Table V. It is clear from the values reported in this table that the difference between (a) the %-errors attained by the FR policy and (b) the %-errors that are attained by the other scheduling policies, is statistically significant. This finding further implies that the "fluid" relaxation developed in this work, and the accompanying logic of Equations 6 and 7, manage to capture effectively the basic workflow dynamics of the considered CRLs that shape their performance.

Finally, we also report that the largest LP formulations resulting from the "fluid" relaxation of the CRL configurations of Table III were solved in less than 3 seconds. Hence, unless the scale of the processing times involved is very small (i.e., in the order of a few seconds), the presented methodology will be very comfortably implemented in the context of the CRLs that are considered in this experiment. But in the next section, we also consider more explicitly the scaling of the LP solution times as the size of the underlying CRL increases.

B. Demonstrating and assessing the tractability of the presented method

In this section we report and discuss the results from an additional set of numerical experiments that intended to investigate empirically the increase of the solution time of the proposed LP relaxation as the underlying CRLs are scaled up to some pretty sizable configurations. These results are reported in Table VI.

More specifically, the considered LP relaxation was formulated and solved for 20 configurations, with each configuration being defined by the first two columns of Table VI. In particular, the first column of this table reports the number of workstations of the corresponding configurations, the buffer size of each workstation, and the mean processing time of the involved processing stages. On the other hand, the second column of Table VI reports the number of times that the line is traversed by each part, and therefore it also determines the number of stages of the corresponding process plan. Finally, the third column of Table VI reports the solution times for the corresponding LP relaxations; for each of the 20 CRL

TABLE IV: An empirical characterization of the performance that is attained by the various heuristic policies considered in the experiment of Section IV-A.

Config.	% error	FR	FBFS	LBFS	SPT-FBFS	SPT-LBFS	MP
Conf 1	Avg.	0	2.323356	0	1.049778	0.581273	0
	Min.	0	0.230038	0	0	0	0
	Max.	0	7.453339	0	3.863308	3.486496	0
Conf 2	Avg.	0	2.563791	0	1.24137	0.765513	0
	Min.	0	0.02166	0	0	0	0
	Max.	0	7.803391	0	4.640675	3.682091	0
Conf 3	Avg.	0.558299	4.281953	0.986135	2.446898	2.3089	1.580789
	Min.	0	0.00912	0	0.00048	0	0.00112
	Max.	1.604902	12.173708	2.566975	5.517447	4.551116	3.45924
Conf 4	Avg.	0.424129	2.53043	0.116397	0.893047	0.868761	1.627846
	Min.	0	0	0	0	0	0
	Max.	1.603027	11.14905	0.503845	3.707549	3.707549	6.765885
Conf 5	Avg.	0.056252	1.072059	0.000941	0.250837	0.189375	0.845842
	Min.	0	0	0	0	0	0
	Max.	0.627573	8.386695	0.011211	1.843072	1.248081	5.951457
Conf 6	Avg.	0.787126	3.087116	0.424329	1.501644	1.074064	2.058273
	Min.	0	0.045081	0.000827	0.00506	0.000827	0.01904
	Max.	2.468721	10.00878	1.711332	6.370564	4.164482	7.04788
Conf 7	Avg.	0.262069	2.921857	2.242041	2.497875	2.538068	1.514924
	Min.	0	0.0143	0.009941	0.0143	0.0147	0.045611
	Max.	1.077797	9.432003	7.589988	8.149082	7.843766	5.124153
Conf 8	Avg.	0.348257	2.601589	1.797056	2.174541	2.152668	2.067119
	Min.	0.00012	0.00948	0.00022	0.00948	0.01296	0.013749
	Max.	1.365613	12.276351	9.255655	7.782629	7.782629	8.948562
Conf 9	Avg.	0.074912	1.331306	0.341868	0.668099	0.640542	0.586824
	Min.	0	0	0	0	0	0
	Max.	0.353171	9.699296	2.322959	4.391537	4.391537	5.283458
Conf 10	Avg.	3.318123	5.666601	4.888519	5.035612	4.983614	4.061828
	Min.	0.160619	0.781178	0.372858	0.954592	0.94859	0.445644
	Max.	6.94834	14.570786	12.520645	15.677988	15.684631	9.899285
Conf 11	Avg.	0.837663	2.777581	3.677889	1.303101	1.69863	1.633319
	Min.	0.032066	0.017783	0.257924	0.017783	0.092037	0.161028
	Max.	1.800238	12.551314	10.558961	4.213472	7.117571	4.232176
Conf 12	Avg.	1.121402	1.750244	2.452877	0.513728	0.583221	2.630212
	Min.	0.008642	0.032489	0.001541	0.001401	0.001541	0.019846
	Max.	2.579237	6.878605	8.388345	0.984445	1.650506	8.486197
Conf 13	Avg.	1.153944	2.264382	4.354457	2.158219	2.663196	2.176783
	Min.	0.034693	0.056632	0.222106	0.158161	0.2213	0.030272
	Max.	2.823261	8.684196	10.984664	8.385657	9.550511	5.152178
Conf 14	Avg.	1.330487	4.610859	1.528115	2.281134	2.089135	3.060204
	Min.	0.009692	0.047781	0.058834	0.058834	0.059004	0.099643
	Max.	2.752433	8.670543	4.278931	4.618316	5.159253	5.3891
Conf 15	Avg.	1.087259	2.441627	3.636966	0.92349	1.102124	2.274138
	Min.	0.484144	0.448128	1.277105	0.143185	0.143185	0.662133
	Max.	2.075733	9.279909	10.152834	3.754415	3.754415	5.461083
Conf 16	Avg.	1.485586	2.141188	3.186278	0.876134	1.005724	2.443547
	Min.	0.077542	0.025837	0.43741	0.025837	0.025837	0.195422
	Max.	3.165834	7.171983	8.032857	2.690495	2.823853	4.837758
Conf 17	Avg.	1.23399	3.190605	6.283274	3.711618	4.010187	3.144512
	Min.	0	0.490636	0.030681	0.490636	0.030681	0.067602
	Max.	3.581841	6.302036	15.98632	7.48842	7.48842	7.174882
Conf 18	Avg.	3.431575	8.666107	10.35568	8.993217	8.655773	11.74234
	Min.	1.294327	6.396949	6.432713	6.396949	6.432713	7.038907
	Max.	4.468809	11.394275	16.312115	12.035385	10.939199	16.799726
Conf 19	Avg.	3.372908	8.528441	10.312104	8.869568	8.533083	11.517266
	Min.	1.07888	5.816134	6.021595	5.816134	5.638849	7.232241
	Max.	4.546285	10.8824	16.257885	11.561644	10.519218	16.014564
Conf 20	Avg.	3.35003	8.658793	10.693207	9.087125	8.751012	11.758839
	Min.	0.77972	5.625051	8.035321	6.695957	6.205683	7.809635
	Max.	5.313402	12.155388	16.182583	12.155388	11.16347	17.40793

TABLE V: A statistical comparison of the performance of the proposed scheduling methodology to the performance of the other heuristic policies considered in the experiment of Section IV-A.

Method	FBFS	LBFS	SPT-FBFS	SPT-LBFS	MP
t-test	4.405229e-48	1.071112e-16	4.142378e-12	5.152820e-11	9.354007e-22
w-test	5.339791e-52	6.755720e-15	4.400029e-14	1.003275e-12	2.285927e-35

configurations, the LP relaxation was formulated and solved at 10 randomly selected decision states, and this column of Table VI reports the minimum, maximum and average values of the corresponding solution times. We also notice that the formulated LPs were solved through the CPLEX Studio IDE 12.8 package, that was running on a Windows 10 computational platform with an Intel Core i5, 2.2 GHz, 2-core processor, and 8GB DDR3 memory.

As it can be seen in Table VI, the solution times for the proposed LP relaxation can be in the order of a few seconds even for some pretty large configurations. This fact is especially true as long as the "spread" of the mean processing times involved is quite small; the corresponding cases are those in blocks #1 and #3 of Table VI.

TABLE VI: An empirical characterization of the computational tractability of the proposed scheduling method.

Basic configuration	# of passes	LP sol. time (sec) (min, mean, max)
5 workstations $B_l = 5,\ \forall l$ $\tau_j = 1,\ \forall j$	2 (10 stages)	(0.23, 0.27, 0.34)
	3 (15 stages)	(0.34, 0.37, 0.42)
	4 (20 stages)	(0.46, 0.50, 0.59)
	5 (25 stages)	(0.71, 0.82, 0.91)
	6 (30 stages)	(0.93, 1.06,1,34)
5 workstations $B_l = 5,\ \forall l$ $\tau_1 = 1$ $\tau_j = 10,\ \forall j \neq 1$	2 (10 stages)	(7.71, 8.16, 8.74)
	3 (15 stages)	(15.53, 16.51, 17.76)
	4 (20 stages)	(26.52, 27.51, 28.42)
	5 (25 stages)	(36.33, 40.14, 44.22)
	6 (30 stages)	(59.98, 59.29, 60.72)
20 workstations $B_l = 5,\ \forall l$ $\tau_j = 1,\ \forall j$	2 (40 stages)	(0.47,0.52,0.70)
	3 (60 stages)	(1.30, 1.46, 1.71)
	4 (80 stages)	(1.67, 1.80, 2.05)
	5 (100 stages)	(2.42, 2.55, 2.69)
	6 (120 stages)	(2.81, 3.12, 3.42)
20 workstations $B_l = 5,\ \forall l$ $\tau_1 = 1$ $\tau_j = 10,\ \forall j \neq 1$	2 (40 stages)	(31.46, 34.25, 38.20)
	3 (60 stages)	(56.94, 68.72, 80.39)
	4 (80 stages)	(95.39, 99.58, 106.38)
	5 (100 stages)	(136.19, 142.29, 151.85)
	6 (120 stages)	(189.25, 197.34, 203.49)

On the other hand, the results of Table VI also suggest that the considered solution times can be severely impacted by a larger "spread" among the underlying processing times. This can be seen by juxtaposing the results of block #2 to those of block #1, and similarly, the results of block #4 to those of block #3. In particular, the configurations of block #4 in Table VI correspond to a "worst-case" scenario where the underlying CRL is pretty sizeable in terms of numbers of workstations and processing stages, and at the same time, all mean processing times involved having very large values except for one. Then, as indicated by the provided formulae in Section III-C, the resulting LPs will be pretty large in terms of the numbers of variables and constraints involved, and therefore, their solution times might scale up to the order of a few minutes. Whether these solution times will be tolerable or not, will depend on the magnitude of the actual processing times involved. If these times are also pretty small, then it might be necessary to control further the computational overhead that is incurred by the presented method, by employing some of the mechanisms that were suggested in the closing part of Section III-C.

V. Conclusions

This paper has extended a scheduling methodology that is based on the solution of a pertinent "fluid" relaxation of the addressed scheduling problem, to complex RAS with blocking and deadlocking effects. For better specificity, the results were presented in the operational context of a re-entrant line with finite buffering capacity at each workstation. This class of re-entrant lines is characterized as "capacitated re-entrant lines (CRL)", and the particular scheduling problem that was addressed in the CRL context, is the maximization of the long-term throughput. But the presented methodology can be adapted to other classes of sequential resource allocation systems (RAS) with blocking and deadlocking effects similar to those studied in [6], and it can also consider additional performance indices, like expected cycle times for the processed

parts, and inventory concentrations at various segments of the line.

From a more methodological standpoint, the presented approach to the considered scheduling problem is substantially enabled and facilitated by a pre-established ability to control the underlying resource allocation for deadlock freedom, and by the further ability to express the corresponding DAP as a set of linear inequalities on the system state. At the same time, the presented developments differ considerably from past implementations of the considered scheduling method, since they must effectively address the blocking and deadlocking effects that take place in the considered CRLs.

Finally, the efficacy of the presented developments has been demonstrated and assessed through extensive numerical experimentation that compared, for a set of "benchmark" CRLs, the performance of the scheduling policies obtained through the presented method, to (i) the performance of the corresponding optimal scheduling policies, and also to (ii) the performance of some other heuristic scheduling policies for these systems that were adapted from the relevant literature for the more traditional re-entrant lines. An additional set of experiments demonstrated and assessed the scalability of the presented method for larger CRL configurations. The results obtained from these experiments suggest that, when combined with the existing LES theory for the considered RAS, the proposed "fluid" relaxation–based method provides an effective instrument for obtaining near-optimal scheduling policies for the considered CRLs, while maintaining the computational tractability that is necessary for the real-time operation of these systems.

Our future work will seek to develop a more structured characterization of the dynamics that are encoded in the proposed "fluid" relaxation, by means of some formal, DES-theoretic modeling frameworks. Besides its inherent theoretical interest, this characterization will enable the further analysis of the structure of the optimal solutions of the corresponding LP, and thus, it can lead to more informed decisions regarding the selection of important parameters like the employed time horizon T for the LP formulation. It will also facilitate the extension of the presented method to more complex RAS classes and/or additional performance considerations, as discussed in the previous paragraphs; these extensions are also part of our ongoing investigations. Finally, an additional line of our future work will seek to bring into the aforementioned analysis the notion of "robustness" that has been pursued w.r.t. such "fluid" relaxations in [20].

REFERENCES

[1] M. Pinedo. *Scheduling*. Prentice Hall, Upper Saddle River, NJ, 2002.

[2] T. Morton and D. W. Pentico. *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*. Wiley, N.Y, N.Y, 1993.

[3] S. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, Cambridge, UK, 2008.

[4] W. M. Wonham. Supervisory control of discrete event systems. Technical Report ECE 1636F / 1637S 2013-14, Electrical & Computer Eng., University of Toronto, 2006.

[5] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems (2nd ed.)*. Springer, NY, NY, 2008.

[6] S. Reveliotis. Logical Control of Complex Resource Allocation Systems. *NOW Series on Foundations and Trends in Systems and Control*, 4:1–224, 2017.

[7] S. A. Reveliotis. *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. Springer, NY, NY, 2005.

[8] J. Y. Choi and S. A. Reveliotis. Relative value function approximation for the capacitated re-entrant line scheduling problem. *IEEE Trans. on Automation Science and Engineering*, 2:285–299, 2005.

[9] R. Li and S. Reveliotis. Performance optimization for a class of generalized stochastic Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 25:387–417, 2015.

[10] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[11] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. 2 (4th ed.)*. Athena Scientific, Belmont, MA, 2012.

[12] J. Y. Choi. *Performance Modelling, Analysis and Control of Capacitated Re-entrant Lines*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2004.

[13] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[14] R. Li and S. Reveliotis. Designing parsimonious scheduling policies for complex resource allocation systems through concurrency theory. *Discrete Event Dynamic Systems: Theory and Applications*, 26:511–537, 2016.

[15] R. Li. *Performance Optimization of Complex Resource Allocation Systems*. PhD thesis, ISyE, Georgia Tech, Atlanta, GA, 2016.

[16] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, NY, NY, 2003.

[17] G. Weiss. On optimal draining of re-entrant fluid lines. *IMA volumes in mathematics and its applications*, 71:91–91, 1995.

[18] G. Weiss. Scheduling and control of manufacturing systems-a fluid approach. 37:577–586, 1999.

[19] T. Bountourelis and S. A. Reveliotis. Optimal node visitation in stochastic digraphs. *IEEE Trans. on Automatic Control*, 53:2558–2570, 2008.

[20] D. Bertsimas, E. Nasrabadi, and I. Ch. Paschalidis. Robust fluid processing networks. *IEEE Trans. on Automatic Control*, 60:715–728, 2015.

[21] P. R. Kumar. Re-entrant lines. *Queueing Systems*, 13:87–110, 1993.

[22] P. R. Kumar. Scheduling semiconductor manufacturing plants. *IEEE Control Systems Magazine*, 14–6:33–40, 1994.

[23] J. G. Dai. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Annals of Applied Probability*, 5:49–77, 1995.

[24] J. G. Dai. Stablity of fluid and stochastic processing networks. Technical Report Miscellanea, No. 9, Dept. of Mathematical Sciences, University of Aarhus, Denmark, 1998.

[25] J. G. Dai and G. Weiss. Stability and instability of fluid models for reentrant lines. *Mathematics of Operations Research*, 21:115–134, 1996.

[26] P. R. Kumar and S. P. Meyn. Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies. *IEEE Trans. Autom. Control*, 41:4–17, 1996.

[27] S. P. Meyn. Stability and optimization of multi-class queueing networks and their fluid models. *Lectures in Applied Mathematics*, 33:175–199, 1997.

[28] S. H. Lu and P. R. Kumar. Distributed scheduling based on due dates and buffer priorities. *IEEE Trans. on Aut. Control*, 36:1406–1416, 1991.

[29] P. R. Kumar. Scheduling manufacturing systems of re-entrant lines. In D. D. Yao, editor, *Stochastic Modeling and Analysis of Manufacturing Systems*, pages 325–360. Springer-Verlag, 1994.

[30] S. A. Reveliotis. The destabilizing effect of blocking due to finite buffering capacity in multi-class queueing networks. *IEEE Trans. on Autom. Control*, 45:585–588, 2000.

[31] J. Y. Choi and S. A. Reveliotis. A generalized stochastic Petri net model for performance analysis and control of capacitated re-entrant lines. *IEEE Trans. on Robotics and Automation*, 19:474–480, 2003.

[32] P. Singer. The driving forces in cluster tool development. *Semiconductor International*, July '95:113–118, 1995.

[33] H. Chen and D. D. Yao. *Fundamentals of Queueing Networks*. Springer, NY, NY, 2001.

[34] M. A. Lawley and S. A. Reveliotis. Deadlock avoidance for sequential resource allocation systems: hard and easy cases. *Intl. Jrnl of FMS*, 13:385–404, 2001.

[35] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modeling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1994.

[36] P. Glasserman and D. Yao. *Monotone Structure in Discrete-Event Systems*. John Wiley & Sons, Inc., NY,NY, 1994.

[37] J. G. Dai and W. Lin. Maximum Pressure Policies in Stochastic Processing Networks. *Operations Research*, 53:197–218, 2005.

[38] P. Rowe. The paired t-test. *Essential Statistics for the Pharmaceutical Sciences, Second Edition*, pages 163–175.

[39] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.