doi:10.1080/00207543.2018.1431417 • www.stefancreemers.be • info@stefancreemers.be

Dynamic order acceptance and capacity planning in a stochastic multi-project environment with a bottleneck resource

Philipp Melchiors
Roel Leus
Stefan Creemers
Rainer Kolisch

Abstract - **We study the integration of order acceptance and capacity planning in multi-project environments with dynamically arriving projects. We model this planning problem as a continuous-time Markov decision process to determine long-term optimal decisions. We examine whether macro-process planning should be performed before or after order acceptance. We characterize the structure of optimal policies, and explore the dependence on a number of parameters such as project payoff, project cost, and order arrival time. We also look into the effects of setup costs and the use of non-regular capacity.**

Keywords - **order acceptance, capacity planning, multi-project scheduling, Markov decision processes, dynamic arrivals**

1 Introduction

In multi-project environments such as Engineer-to-order (ETO) [5] or research and development (R&D) [1, 51], new projects are arriving dynamically. Typically, their content is uncertain and cannot be fully specified in advance. A common problem is the lack of integration between Order Acceptance (OA), which is typically decided by the sales department, and capacity planning, which is usually done by the engineering or production department [60]. While the sales department tends to try to boost revenues by accepting as many projects as possible, R&D and/or production often struggle with congestion at highly utilized resources that are shared by many projects. Long project lead times and unmet due dates are the result.

Multi-project management often adopts a hierarchical approach to planning because the time scale and complexity of the projects can be considerable. As outlined by Hans et al. [20], a hierarchical planning process performs a preliminary analysis upon arrival of an order/project, which is referred to as Macro-Process Planning (MPP). In MPP, a number of characteristics of the order are established, e.g., the approximate work content of specific work packages, the most important precedence constraints, aggregate resource requirements, etc. Obviously, MPP is already a complex task, which usually has to be performed under considerable urgency: potential customers expect a quick response as to whether or not their order is accepted. If MPP is performed after OA, there is less urgency, and costs can be saved. In this case, however, the output of MPP is not available when making the decision whether or not to accept an order.

The results of MPP are used in a Rough-Cut Capacity Planning (RCCP) step, where macro activities (work packages) are allocated to resources that are typically larger organizational units such as departments. One important aspect of the capacity plans is that they help to assess the consequences of a new order with respect to resource utilization, and thus support optimal OA decisions. To the best of our knowledge, the incorporation of the effect of possible future arrivals of new orders into the planning process has not yet been studied.

This work has been inspired by a collaboration with a large supplier of automotive components, with multiple R&D departments. We worked together with the department in charge of modification projects for combustion engine control units. Requests for such projects arrive from internal and external customers. The company operates the R&D department in a market setting where internal orders don't have to be processed but can be outsourced instead; at the same time the department also provides engineering services for external customers. Hence, the department has the option to accept or reject order requests. The department employs a number of specialized resources, in particular one piece of testing equipment which basically acts as the main bottleneck resource and thus determines the flow time of projects.

The company contacted us because due to an increasing demand for modifications of combustion engine control units, the arrival rate as well as the revenue of projects had increased and the R&D department tended to accept too many orders which led to overly long flow time of projects and to too many projects in the system. After initial discussions on how to approach the planning problem, the company explicitly opted for a planning approach where each project would be aggregated to the work which had to be done on the testing equipment. A detailed approach where each project would be depicted as an activity network was soon discarded, because the department already had project scheduling software in use for detailed planning of project activities of accepted projects. This software, however, did not support OA decisions, nor choices regarding MPP before or after OA.

The contributions of this article are threefold. Firstly, after a literature review, we model the integrated order acceptance and capacity planning problem as a Continuous-Time Markov Decision Process (CTMDP). The basic model allows to determine long-term optimal decisions for OA and RCCP. Furthermore, we examine whether MPP should be performed before OA such that OA can take full advantage of the information obtained from MPP. Alternatively, MPP is performed afterwards, such that only basic information from the order is available for OA. As a second contribution, we characterize optimal policies and explore their dependence on a number of parameters such as project payoff, project cost, order arrival time, etc. Finally, we propose two extensions to the basic model. A first extension enables the use of non-regular capacity such as overtime to accelerate the processing of projects. A second extension allows for the incorporation of setup costs. Even though our models do not capture all complexities of a real-world multi-project environment, they provide insight in the dynamics of OA and MPP. These insights have been used by one of the R&D departments of a large supplier of automotive components.

2 Literature review

The relevant literature draws from two related fields: capacity planning and order acceptance. In what follows, we use the term dynamic to indicate that orders/projects arrive over time (analogously to the stochastic knapsack problem [28]). The term static, by contrast, refers to the setting where all projects are available at the start of the planning horizon.

2.1 Project scheduling and capacity planning

The literature on project scheduling deals with the scheduling either of a single project or of multiple projects. In this work we look into multi-project planning, and so we do not cover the single-project literature (for a survey, see [42]). Scheduling multiple projects under uncertainty has been studied by various authors (see [20] for an overview). The literature can be partitioned along two different dimensions of the underlying optimization problem: (1) static vs. dynamic, and (2) deterministic vs. stochastic. Below we focus on stochastic multi-project scheduling. For static deterministic models for job prioritization and order release, see for instance Riezebos [48].

For the static stochastic case, an important class of problems are *multi-armed-bandit problems* (see [17, 43] for a description). Gittins and Jones [17] were the first to show that the optimal policy is a *priority index rule*. For each project, a priority index rule computes an index value that depends only on the state of the project at the given decision time and does not depend on the state of other projects. The indices are used for prioritizing the projects (a project with highest index is selected). An important extension of the multi-armed bandit problem is the *restless bandit problem*, which was first considered by Whittle [57]. For the restless bandit problem, the optimal policy is not necessarily a priority index rule anymore. Kavadias and Loch [26] schedule multiple projects at a single bottleneck resource under more general assumptions than the multi-armed bandit case, such that it becomes a restless bandit problem. They find that under specific conditions the optimal policy is again a priority index rule.

Scheduling in dynamic and stochastic multi-project environments has mostly been considered in the context of queueing networks. Dependent on the models' assumptions, different policies have been shown to be optimal for the special case of queueing systems that consist of a single station. Cox and Smith [8] show that the $c\mu$-policy is optimal for an $M/G/1$ system with different project types to minimize average cost per unit time, where c denotes the holding cost per unit of time of a project type and μ the service rate for the respective project. For the more general $M/G/1$ system with feedback, i.e., where projects can revisit the system after being served while changing type with a given probability, Klimov [30] proves that a priority index rule is optimal.

Adler et al. [1] and Levy and Globerson [32] adapt classic queueing networks to a project setting by allowing queues to have multiple successors (by using forks) and multiple predecessors (by using joins). Based on these models, Cohen et al. [7], as well as Melchiors and Kolisch [38], analyze the performance of priority rules using simulation. Melchiors and Kolisch [39] also propose a CTMDP for the computation of optimal scheduling policies. From a more practical perspective, Hutter et al. [23] describe how they successfully implemented an order-release mechanism in a job-shop-like environment by limiting the workload in the manufacturing system and thus controlling flow times.

Models dedicated to tactical RCCP comprise both scheduling decisions as well as decisions related to capacity deployment. RCCP in static and deterministic environments has been considered by De Boer [10] and Hans [19], for example. In their models, extra "non-regular" capacity can be used in order to process more project activities at a time, or to reduce the duration of an activity. The latter usage has also been referred to as *crashing* and has been considered by many different authors (e.g., [27, 4, 49]). In dynamic stochastic environments, activity crashing is closely related to control of services rates. Crabill [9] investigates the optimal control of service rates for an $M/M/1$ system, and finds that an optimal policy is characterized by multiple thresholds on the number of projects in the system. When a threshold is exceeded, the service rate switches to its next higher value.

Scheduling in the presence of setup times and costs was first considered by Hofri and Ross [22] for a queueing system with two customer classes. This case was extended to an arbitrary numbers of customer classes by Liu et al. [33]. Reiman and Wein [47] apply a heavy-traffic approximation to obtain near-optimal policies for this case.

2.2 Order acceptance

Order acceptance problems have been considered in static as well as dynamic environments. In static environments, a set of projects is given at the beginning of the planning horizon, from which a subset has to be selected so as to optimize an objective subject to a set of constraints. Consequently, OA is mostly referred to as *selection* in static environments. Models based on mathematical programming have been proposed by Bard et al. [3] and Loch et al. [35]. Loch and Kavadias [34] present a more aggregate model that considers the problem from a financial perspective without explicit consideration of projects as discrete items. Joint optimization of OA and scheduling decisions in a static context has been considered by Slotnick and Morton [52] and Talla Nobibon and Leus [54]. For an illustration of how to integrate order acceptance into aggregate planning for standardized items, we refer to Brahimi et al. [6].

OA for dynamic environments has been studied in different fields. In multi-project planning, different models exist that are generalizations of the Dynamic Stochastic Knapsack Problem (DSKP). A number of variants of the DSKP have been proposed by Ross and Tsang [50] and Kleywegt and Papastavrou [28, 29]. The basic idea is as follows: given a single resource with limited capacity, schedule items that arrive dynamically with stochastic interarrival times and varying demand for capacity. If all resources are in use then the item must be rejected (i.e., queueing is not allowed).

Perry and Hartman [44] look into OA with a single resource for multiple periods and for projects that consist of only a single activity. As the capacity is limited in each period, the problem can be dealt with as a multi-knapsack problem. Herbots et al. [21] extend the model of Perry and Hartman [44] by allowing more complex resource allocation schemes, which are typical for RCCP. Although both models are able to take into account stochastic interarrival times, only deterministic project durations are considered.

OA with stochastic interarrival times and stochastic project durations has been examined in queueing theory; projects are again typically modeled as only one activity. One of the first papers is by Naor [41], who considers OA decisions for an $M/M/1$ system with a single project type. A holding cost is incurred per unit of time a project is in the system, and payoffs are obtained for each accepted project. The average reward is to be maximized. Generalizations of the basic model to general distributions and multiple capacity units appear in Yechiali [58, 59], Knudsen [31], and Feinberg and Yang [16]. Feinberg and Yang [16], in particular, consider an $M/M/c$ system with multiple project types where holding costs, arrival rates, and payoffs vary between the types, while all project types have the same expected duration (service rate). For the systems considered, OA policies are monotone, meaning that an order of each type is accepted until the number of projects in the system exceeds a threshold that may depend on the project type. However, scheduling decisions and decisions regarding resource capacities have not been included into the optimization. Typically, scheduling decisions are made following a FCFS rule.

Joint optimization of OA and scheduling decisions in queueing theory is studied by De Serres [11, 13]. The author considers an $M/M/1$ system with two project types where the expected duration depends on the type, and preemption of projects already in process is allowed. Based on extensive experimental studies he finds that, for many cases, the optimal policies exhibit a monotone structure with respect to the OA decisions. The author does not, however, provide formal proofs for this structure of optimal policies. Furthermore, although in many cases the $c\mu$-policy has been shown to be optimal for scheduling decisions, De Serres [13] provides a counterexample when OA is included. Although the work of De Serres comes closest to our problem, it neglects important issues relevant for OA in multi–project organizations. Firstly, no decisions with respect to resource capacities are taken into account. Secondly, it is assumed that all project-related information (except the duration) is fully known upon arrival. Thirdly, the costs of performing MPP before OA are not taken into account.

The joint optimization of OA and scheduling has also been investigated on a heuristic basis. Wester et al. [56] and Van Foreest et al. [55] develop and test (via simulation) different OA and scheduling heuristics for a system that consists of a single resource that processes one project at a time. The heuristics are characterized by different levels of detail concerning the system information used. The case with multiple resource types and projects consisting of multiple activities has been considered by Ebben et al. [15], Ivanescu et al. [24], and Ivanescu et al. [25]; the latter two references also allow for stochastic activity durations. A first version of the model presented in this paper without considering setup costs and idleness is available in Melchiors [37].

3 Basic model

In the remainder of the text, the term *order* refers to a request from a customer before OA, while a *project* is a request that has been accepted.

3.1 Multi-project environment

Following Adler et al. [1], we focus on the case where projects, although they are unique, have enough similarities to be categorized into project types $p \in \mathcal{P}$. All projects of type p arrive according to a stochastic Poisson arrival process with rate λ_p. On completion, a payoff y_p is obtained. For simplicity of the analysis, we consider only a single resource that can be seen as the *bottleneck resource* of the system. This bottleneck has unit capacity: only one project can be processed at a time (see [26] for a similar modeling choice). Detailed information on individual activities, such as their resource usage, becomes available only after MPP, and we therefore assume that the type to which a project belongs is discovered only *after* MPP has been performed.

In line with most of the literature, we assume that for each project only a single activity is to be processed on the bottleneck resource. Furthermore, we also assume that the duration of the bottleneck activity of a project of type $p \in \mathcal{P}$ is exponentially distributed with rate μ_p; the expected duration is $\overline{d}_p = \frac{1}{\mu_p}$. Without loss of generality, we assume that the duration of the project equals the duration of the bottleneck activity. During execution on the bottleneck resource, a cost w_p^{E} is incurred per unit of time. For the activities that are processed on non-bottleneck resources, a total cost k_p^{E} is incurred. During the stay of the project in the system, a holding cost w_p is incurred per unit of time.

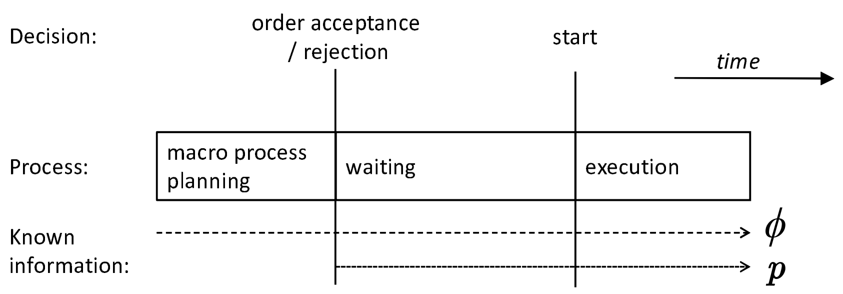
Before performing MPP, less information is available, and only the *parent* project type $\phi \in \Phi$ of a project is known. The parent type is known upon arrival of the project. After MPP has been performed, more information becomes available, and the *child* project type $p \in \mathcal{P}$ of the parent type ϕ becomes known; by $\mathcal{P}_\phi \subset \mathcal{P}$ we will denote the set of child types corresponding to parent ϕ. For the case of the engineering department discussed in the introduction, is is known from the outset if the order request is internal or external and what modification of the control unit is requested. However, MPP is necessary in order to translate the modification request in an estimate for the overall time the testing equipment is needed as well as an estimate on further work not operated on the test equipment. The arrival rate of projects of parent type ϕ is $\lambda_\phi = \sum\limits_{p \in \mathcal{P}_\phi} \lambda_p$.

3.2 Order acceptance and resource allocation

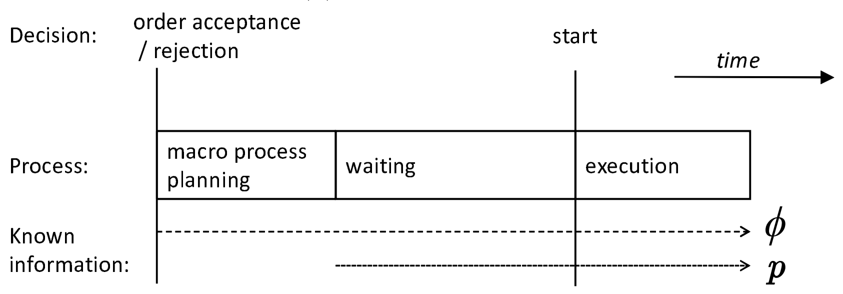
An order arriving at the system may be *accepted* or *rejected* with or without performing MPP beforehand. In the following, we refer to MPP performed before OA as *advanced* MPP and to MPP performed afterwards as *postponed* MPP. Before MPP, only the parent project type ϕ is known; through MPP we discover the child project type $p \in \mathcal{P}_\phi$. Since MPP requires resources, costs (such as labor costs) are incurred. Let $\phi_p \in \Phi$ be the parent type of child project type $p \in \mathcal{P}$. For simplicity, we assume that a fixed cost $k_{\phi_p}^{\mathrm{AM}}$ is incurred for advanced MPP of a project type $p \in \mathcal{P}$, while in the case of postponed MPP a fixed cost $k_{\phi_p}^{\mathrm{PM}}$ is incurred. It is realistic to assume that $k_{\phi_p}^{\mathrm{AM}} > k_{\phi_p}^{\mathrm{PM}}$.

Figure 1 illustrates the difference between *advanced* and *postponed* MPP. Note that we ignore the time spent for MPP itself as it is short relative to the duration of the project and/or the project interarrival time.

To simplify the analysis, we assume that idling the resource is not allowed when there are waiting projects. When a project is completed, we therefore select a new project for execution from the queue of accepted projects waiting for the resource. Furthermore, we assume that only one project may be processed at a time, and that the project in process cannot be preempted. The latter requirement is realistic since preemptions are often detrimental to system performance [18, 2].



(a) Advanced MPP



(b) Postponed MPP

Figure 1: Alternative processes depending on the timing of MPP

3.3 Markov decision process

We model the problem as a CTMDP to derive optimal decisions. At any decision time, the system is fully characterized by its state $s = (\boldsymbol{n}^{\mathrm{W}}, \boldsymbol{n}^{\mathrm{E}})$, where $\boldsymbol{n}^{\mathrm{W}} = \left(n_1^{\mathrm{W}}, n_2^{\mathrm{W}}, \ldots, n_{|\mathcal{P}|}^{\mathrm{W}}\right)$ and $\boldsymbol{n}^{\mathrm{E}} = \left(n_1^{\mathrm{E}}, n_2^{\mathrm{E}}, \ldots, n_{|\mathcal{P}|}^{\mathrm{E}}\right)$ are vectors in which n_p^{W} denotes the number of waiting projects of type $p \in \mathcal{P}$ and n_p^{E} is the number of projects of type $p \in \mathcal{P}$ that are in process. Since we consider only a single resource that cannot process more than one project at a time, the elements in $\boldsymbol{n}^{\mathrm{E}}$ are either 1 (if the project is in process) or 0, otherwise with $\mathbf{1}\boldsymbol{n}^{\mathrm{E}} \leq 1$; i.e., at most one project can be processed on the bottleneck resource at one time. The state space $\mathcal{S}$ is the set of all feasible states s, and state $s^0 = ((0, \ldots, 0), (0, \ldots, 0))$ represents an empty system. We define the set $\mathcal{A}(s)$ of possible decisions for state $s \in \mathcal{S}$. A decision $a = (\boldsymbol{\delta}^{\mathrm{AM}}, \boldsymbol{\delta}^{\mathrm{E}}) \in \mathcal{A}(s)$ entails vector $\boldsymbol{\delta}^{\mathrm{AM}}$, which describes OA decisions, and vector $\boldsymbol{\delta}^{\mathrm{E}}$, pertaining to capacity allocation. Below, we describe these two vectors in more detail.

In order to reduce the size of the state space, we adopt an idea of Ross and Tsang [50], and decide in state $s \in \mathcal{S}$ whether or not we accept an order depending on its project type. In this way, we do not need to store an additional state variable that keeps track of whether an order has arrived or not. Thus, vector $\boldsymbol{\delta}^{\mathrm{AM}} = \left(\delta_1^{\mathrm{AM}}, \delta_2^{\mathrm{AM}}, \ldots, \delta_{|\mathcal{P}|}^{\mathrm{AM}}\right)$ is a vector of binary variables δ_p^{AM} that indicate whether or not the next order is to be accepted:

$$\delta_p^{\mathrm{AM}} = \begin{cases} 1 & \text{then accept the order if it is of type } p, \\ 0 & \text{then reject the order if it is of type } p. \end{cases}$$

We distinguish between the following three settings for OA when an order of type p arrives (at arrival only the parent type ϕ_p is known):

1. If $\delta_{p\prime}^{\mathrm{AM}} = 0$ for all $p\prime \in \mathcal{P}_{\phi_p}$ then the order is rejected immediately, and obviously no MPP is performed. As a result, no costs are incurred and there is no state transition.

2. If $\delta_{p\prime}^{\mathrm{AM}} = 1$ for all $p\prime \in \mathcal{P}_{\phi_p}$ then regardless of the detailed project type, the parent type is accepted, and it is optimal to postpone MPP. A transition occurs to a new state in which n_p^{W} is increased by 1, and we incur cost $k_{\phi_p}^{\mathrm{PM}} + k_p^{\mathrm{E}}$.

3. In all other cases, some $p\prime \in \mathcal{P}_{\phi_p}$ will be accepted and other types rejected, and we perform MPP to discover the detailed project type. We incur cost $k_{\phi_p}^{\mathrm{AM}} + k_p^{\mathrm{E}}$.

In case of postponed MPP, only the parent type ϕ_p will be known at the time of OA.

When a project of type p is completed, a reward y_p is obtained, and a new project can be started. Vector $\boldsymbol{\delta}^{\mathrm{E}} = \left(\delta_1^{\mathrm{E}}, \delta_2^{\mathrm{E}}, \ldots, \delta_{|\mathcal{P}|}^{\mathrm{E}}\right)$ is a vector of binary variables δ_p^{E} that decide which project type is to be dispatched for processing on the resource:

$$\delta_p^{\mathrm{E}} = \begin{cases} 1 & \text{dispatch a project of type } p \text{ to the bottleneck resource,} \\ 0 & \text{otherwise.} \end{cases}$$

An optimal decision corresponds to a non-idling policy (see also Meyn [40]), and makes sure that only one project is processed on the bottleneck resource at any one time. Furthermore, we assume that a project in process cannot be preempted.

Apart from the immediate transition when a project is completed and another one is started, the time spent in any other state $s \in \mathcal{S}$ is exponentially distributed: new orders arrive at rate $\sum\limits_{p \in \mathcal{P}} \lambda_p$, and the ongoing project is completed with rate μ_p. During the time spent in a state s, several costs are incurred. Let $c(s,a)$ denote the cost incurred per unit of time spent in state s with decision a (recall that w_p and w_p^{E} represent the holding cost and the bottleneck execution cost per unit of time):

$$c(s,a) = \sum_{p \in \mathcal{P}} \left(w_p \cdot n_p^{\mathrm{W}}(s) + \left(w_p^{\mathrm{E}} + w_p \right) \cdot n_p^{\mathrm{E}}(s) \right).$$

Define $y(s,a,s')$ to be the reward obtained upon a transition from state s to s' through decision a, which is the payoff y_p or the order acceptance cost (i.e., $k_{\phi_p}^{\mathrm{AM}}$ or $k_{\phi_p}^{\mathrm{PM}}$), depending on the transition. The objective function of a policy π considers the long-term average reward per unit of time:

$$g(\pi) = \liminf_{N \to \infty} \frac{\mathbb{E}\left[\sum\limits_{n=0}^{N-1} y(s_n, \pi(s_n), s_{n+1}) - \sum\limits_{n=0}^{N} c(s_n, \pi(s_n)) \cdot \tau_n \right]}{\mathbb{E}\left[\sum\limits_{n=0}^{N} \tau(s_n, \pi(s_n)) \right]}, \tag{1}$$

where $\mathbb{E}$ is the expectation operator, n is the state index, $\pi(s_n)$ is the decision made in state s_n that is dictated by policy π, τ_n is the stay time in state s_n, and $\tau(s_n, \pi(s_n))$ is the transition time to move from state s_n to state s_{n+1} after decision $\pi(s_n)$ has been made. We then seek an optimal policy $\pi^* = \arg\max\limits_{\pi \in \Pi} \{g(\pi)\}$, where Π is the set of all feasible policies. Further details of the CTMDP are provided in the Appendix.

4 Analysis for the base case

In order to investigate the benefit of accepting orders without MPP and the structure of optimal policies, we consider an over-utilized system where $\sum_{p \in \mathcal{P}} \lambda_p \overline{d}_p \geq 1$. In other words, stability of the system can only be obtained by rejecting some of the incoming orders. In what follows, we assume that there are two child project types $1, 2 \in \mathcal{P}$ of the same parent project type $\phi_p = 1$. Based on our experience in one of the R&D departments of a supplier of automotive components, we define three different cases; the details are provided in Table 1. In all cases, we assume that postponing MPP reduces the cost of MPP by 50%. For cases 1 and 3, the cost of advanced MPP amounts to 10% of the total per-period cost (except holding cost) when all projects are accepted, which is $k_1^{\mathrm{AM}} + \lambda_1 \cdot (w_1^{\mathrm{E}} \cdot \overline{d}_1 + k_1^{\mathrm{E}}) + \lambda_2 \cdot (w_2^{\mathrm{E}} \cdot \overline{d}_2 + k_2^{\mathrm{E}}) = 50$. Given a holding cost $w_p = 1$, a project should remain in the system for at most 20 time units in order to have a positive contribution to the overall profit of the company. For case 2, the cost of MPP is doubled but non-bottleneck execution costs are lower. Except for their expected duration, both project types are identical in cases 1 and 2. In case 3, the two project types also have different execution costs.

In the following analysis we study two different settings. In the first setting, MPP is *regular* or inflexible: it is always performed before OA. In the second setting, MPP is *flexible*, meaning that it can be performed before or after OA. For the first case, Figure 2(a) shows the structure of optimal OA decisions depending on the number of projects in the system: $n_p := n_p^{\mathrm{W}} + n_p^{\mathrm{E}}$, for $p = 1, 2$. The optimal objective function value is g^*. The light grey region indicates that with a sufficiently low number of projects in the system, any project can be accepted without MPP. In the medium grey region, only projects of type 1 should be accepted. In order to distinguish between projects of type 1 and 2, MPP is necessary. In the dark region, all newly arriving projects are rejected (i.e., no MPP is performed). The structure of these OA decisions is in line with De Serres [12], who found that optimal OA decisions are monotone in the number of projects in the system. In order to highlight the benefit of flexible MPP, we compare the results with *inflexible MPP*. By moving from the inflexible to the flexible setting, the average reward per period increases by 12%, and the light grey region becomes slightly larger because the benefit of postponed

Parameter	Case 1	Case 2	Case 3
$(\overline{d}_1, \overline{d}_2)$	$(0.6, 1.4)$	$(0.6, 1.4)$	$(0.6, 1.4)$
(w_1, w_2)	$(1, 1)$	$(1, 1)$	$(1, 1)$
(y_1, y_2)	$(70, 70)$	$(70, 70)$	$(70, 70)$
(λ_1, λ_2)	$(0.5, 0.5)$	$(0.5, 0.5)$	$(0.5, 0.5)$
$(k_1^{\mathrm{E}}, k_2^{\mathrm{E}})$	$(25, 25)$	$(20, 20)$	$(28, 12)$
$(w_1^{\mathrm{E}}, w_2^{\mathrm{E}})$	$(20, 20)$	$(20, 20)$	$(28, 12)$
k^{AM}	5	10	5
k^{PM}	2.5	5	2.5

Table 1: Problem parameters for the three cases

MPP outweighs the benefit of additional information obtained from MPP.

Figure 2(b) shows the results for the second case, where we observe a higher benefit for flexible MPP compared to case 1 (26% higher objective value); the optimal objective value g^* also increases compared to case 1, whereas it decreases for regular MPP. In total, less projects are accepted (because more orders of type 2 are accepted, with higher expected workload), however, more projects are accepted with postponed MPP (because MPP is more expensive, although the processing cost for non-bottleneck resources has decreased). Clearly, the higher the additional cost of advanced MPP (when compared to postponed MPP), the higher the advantage of flexible MPP. Similar observations can be made when different parameters are used; we will use the following definition.

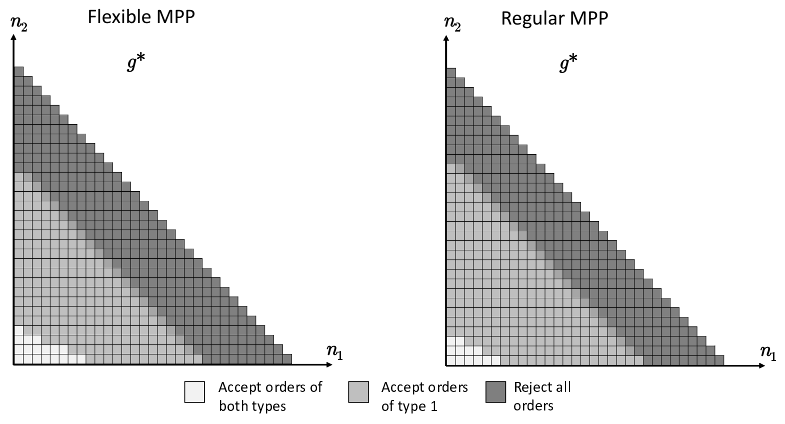
Definition 1. *Be $p_1, p_2 \in \mathcal{P}_\phi$ for some $\phi \in \Phi$. We say that p_1 dominates p_2 if $\overline{d}_1 \leq \overline{d}_2$, $w_{p_1} \leq w_{p_2}$, $w^E_{p_1} \leq w^E_{p_2}$, $k^E_{p_1} \leq k^E_{p_2}$, $y_1 \geq y_2$, and at least one inequality is strict.*

If a dominant project type exists, then typically there will be a set of values (n_1, n_2) for which the dominant type will be accepted and the dominated type rejected. This is in line with the literature on OA, where a threshold of projects is found to be optimal.

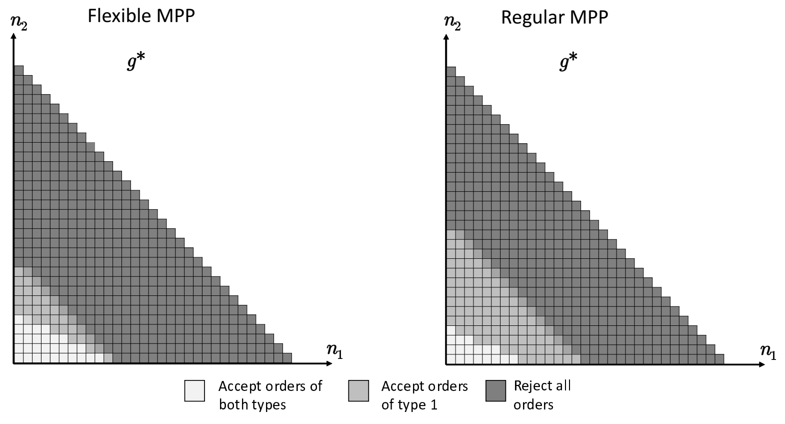
Finally, we consider the third case, where a dominance relationship as described above does not exist. Figure 2(c) shows the corresponding OA decisions. In the third case, the average reward is about 12.7% higher for flexible MPP. MPP is never performed before OA as newly arriving projects of both types are always accepted. When MPP is postponed, we notice that the acceptance region becomes slightly larger. The smaller cost for postponed MPP allows for a higher holding cost without becoming unprofitable.

In all three of the foregoing cases, the allocation of projects to the bottleneck resource is in line with a $c\mu$-policy, which confirms the findings of De Serres [12] for the case with preemptions. We have also shown that the value of information from MPP before OA is used to discriminate between heterogeneous project types. This indicates that the heterogeneity of project types may have an impact on the benefit of flexible MPP. In order to further investigate this potential impact, we have extended the second case and used the following combinations of expected durations: $(1;1)$ (no heterogeneity), $(0.6;1.4)$ (medium heterogeneity) and $(0.2;1.8)$ (high heterogeneity). Figure 3 shows the results. We observe that the benefit of flexible MPP decreases slightly as heterogeneity increases. This can be explained by the fact that MPP is performed more frequently before OA for more heterogeneous projects. Similar observations can be made for heterogeneous payoffs or holding costs, as long as there is a dominance relationship.

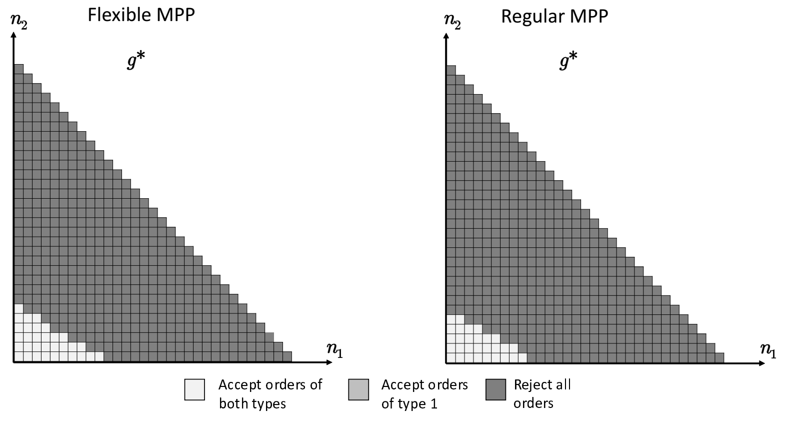
To summarize our findings, we note that postponed MPP may have considerable benefit, depending on the difference in cost with regular MPP. Furthermore, the structure of the OA decisions remains monotone for most cases while the acceptance regions change. As long as there is a dominance relationship between project types, the acceptance region of the dominated project type is contained in the acceptance region of the dominating project type. Finally, in our experiments, optimal resource allocation decisions follow a $c\mu$-policy.



(a) Case 1



(b) Case 2



(c) Case 3

Figure 2: Optimal OA decisions

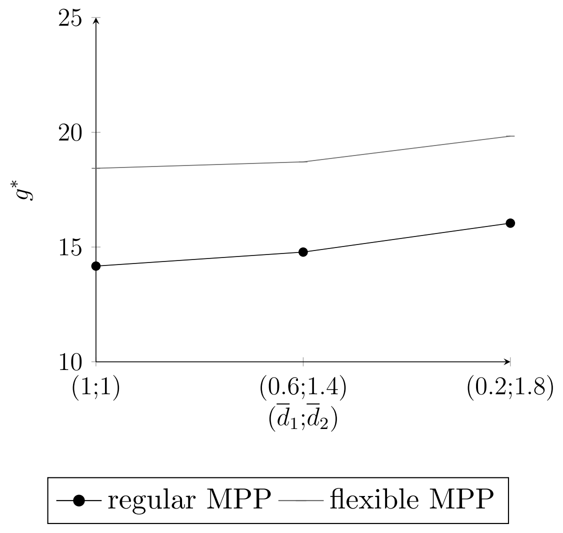


Figure 3: Benefit of postponing MPP

5 Extensions

In this section we consider two extensions of the basic model. First we allow for non-regular capacity to be used to process projects, and second we investigate the impact of setup costs that are incurred when switching from one project type to another. The corresponding generalizations of the CTMDP are commented in Section 6.2 in the Appendix.

5.1 Non-regular capacity

Since we consider a problem at the tactical planning level, we assume that *non-regular* capacity is available [19, 21]. Non-regular capacity may be used to process multiple projects at a time [44], or to *crash* a project, in the sense that it is processed in less time [27]. In the case of the engineering department, engineers can work to some extend longer hours and by this, the duration of the projects is shortened. Hence, we consider the second alternative. Be δ^{C} a continuous variable with $0 \leq \delta^{\mathrm{C}} \leq 1$ describing the amount of non-regular capacity that is invoked to reduce the duration of the project in process at the bottleneck resource. A decision a then takes the form:

$$a = \left(\boldsymbol{\delta}^{\mathrm{AM}}, \boldsymbol{\delta}^{\mathrm{E}}, \delta^{\mathrm{C}}\right). \tag{2}$$

We assume the service rate to be linearly dependent on δ^{C}: the rate corresponds to $\mu_p(1 + z_p \cdot \delta^{\mathrm{C}})$, where z_p is the maximum increase of the service rate, obtained with full usage of the non-regular capacity. Non-regular resource usage incurs additional costs $w^{\mathrm{C}}\delta^{\mathrm{C}}$ per unit of time that a project is processed in "crashed" mode; this is common in many areas such as service organizations [14, 36]. The cost rate $c(s, a)$ per time unit the system is in state s after selecting decision a then becomes:

$$c(s,a) = \sum_{p\in\mathcal{P}} \left(w_p \cdot n_p^{\mathrm{W}}(s) + \left(w_p^{\mathrm{E}} + w_p\right) \cdot n_p^{\mathrm{E}}(s) + w^{\mathrm{C}} \cdot \delta^{\mathrm{C}}(a)\right). \tag{3}$$

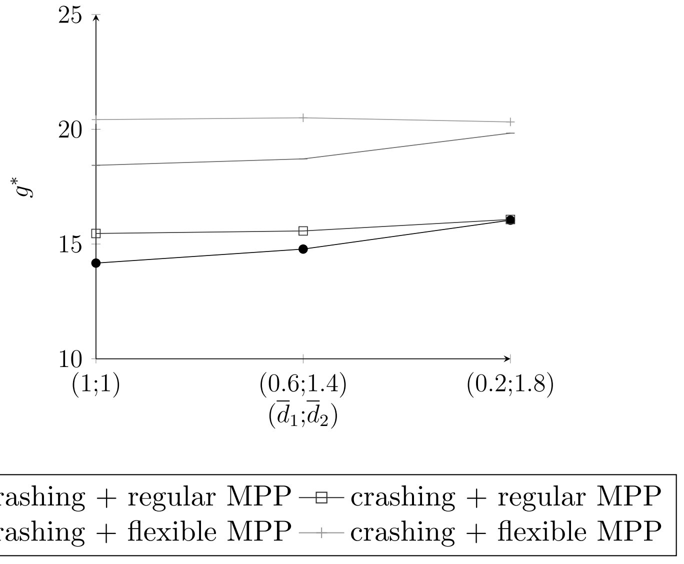


Figure 4: Benefit of flexible MPP and non-regular capacity

It can be shown that the optimal usage of non-regular capacity is either 100% or 0% when the cost and service rates are linear in the amount of non-regular capacity that is deployed. This is in line with a *bang bang* control [53], where the optimal service rate in a given system state is always one of the two extreme values of a feasible interval. Lastly, we assume that the use of non-regular capacity may be changed at any time.

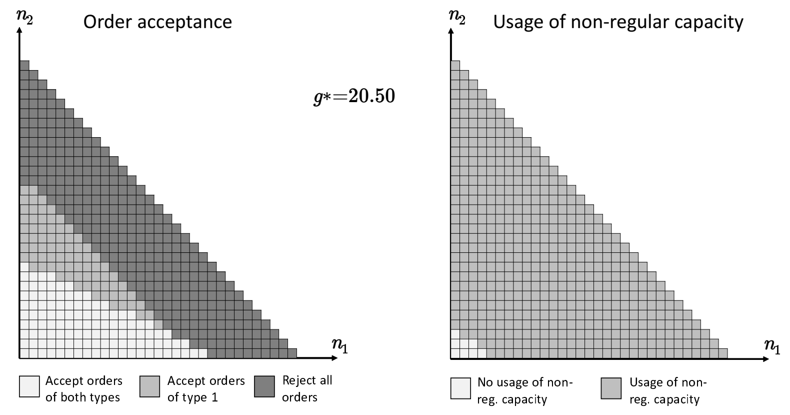
To study the effect of the inclusion of non-regular capacity into the base model, we consider the second base case of Section 4, and we focus on the following research questions:

1. What is the benefit of non-regular capacity, and how does it interact with the benefit of flexible MPP?

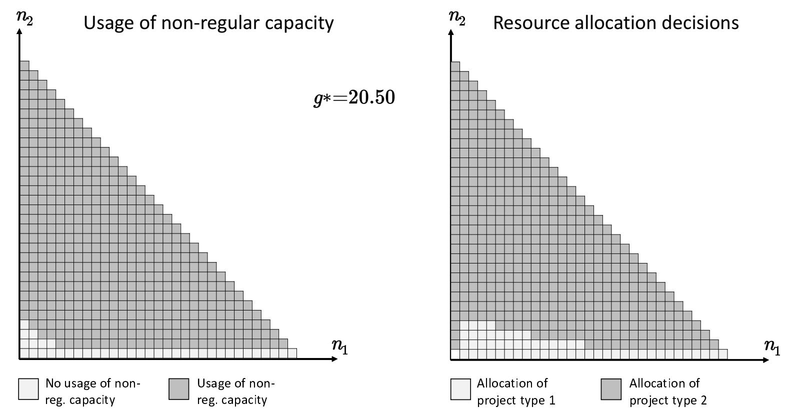
2. What is the impact on the structure of optimal policies?

In order to investigate the benefit of non-regular capacity, we assume that the service rate can be increased by at most 50%, so $z_1 = z_2 = 0.5$; the cost coefficient w^{C} is set to 15. The benefit of non-regular capacity is considered at different levels of project heterogeneity (defined by the difference in expected duration). Figure 4 shows the results when crashing is allowed. The less heterogeneous the project types are, the more non-regular capacity is beneficial. Obviously, with more heterogeneous project types, the OA decisions are more selective, and rule out the less attractive project types. Thus, the expected workload in the system (and hence the need for crashing projects) decreases. Interestingly, non-regular capacity becomes more beneficial when flexible MPP is allowed as the option of postponing MPP makes OA less selective. We have obtained similar observations when payoffs or execution costs are varied.

Next, we analyze the optimal structure of the optimal policy for the second case. Figure 5(a) shows the optimal decisions with respect to OA and non-regular capacity. Comparing the OA decisions with those in Figure 2(b), we find that the policy becomes less selective



(a) Case 2



(b) Variant of case 2 with $z_1 = 0$ and $z_2 = 1.5$

Figure 5: Optimal OA decisions and resource allocation with non-regular capacity

as more projects are accepted overall, while the basic structure of the OA decisions is very similar. Non-regular capacity is only used if the number of projects exceeds a threshold. The more projects are in the system, the higher the pressure to accelerate the processing of projects in order to reduce holding costs. Finally, we note that the resource allocation decisions are again in line with a $c\mu$-policy.

We have verified these observations in a number of additional numerical experiments. For most cases, we find that resource allocation no longer follows the classic $c\mu$-policy because crashing may lead to a shorter expected duration. In general, we see that when using non-regular capacity, project type p_2 is preferred over project type p_1 if:

$$w_{p_1}\mu_{p_1} < w_{p_2}\mu_{p_2}(1 + z_{p_2}). \tag{4}$$

To demonstrate this, we set $z_1 = 0$ and $z_2 = 1.5$ for the second case. Thus, we have:

$$\frac{1}{0.6} = 1.667 < \frac{1}{1.4} \cdot 2.5 = 1.786. \tag{5}$$

Figure 5(b) shows the corresponding optimal decisions. We observe that it is almost always preferable to employ non-regular capacity, except when there are very few projects of type 2 (and in fact, when there are no projects of type 2 then non-regular capacity is never used). This can be explained as follows. In our definition of a state, we do not take into account which project type is currently in process. Thus, if project type 1 is preferred in some state, there is another state with the same number of projects in the system where a project of type 2 is still in process. Hence, it makes sense to use non-regular capacity.

Our numerical experiments have shown that these findings also hold for different cases and different parameter settings. Therefore, we extend Definition 1 by adding z_p.

Definition 2. *Be $p_1, p_2 \in \mathcal{P}_\phi$ for some $\phi \in \Phi$. We say that p_1 dominates p_2 pointwise if $\overline{d}_1 \leq \overline{d}_2$, $w_{p_1} \leq w_{p_2}$, $w^E_{p_1} \leq w^E_{p_2}$, $k^E_{p_1} \leq k^E_{p_2}$, $y_1 \geq y_2$, $z_1 \geq z_2$, and at least one inequality is strict.*

5.2 Setup cost

In the literature on order acceptance and capacity planning, setup costs have been studied by Wester et al. [56] and Van Foreest et al. [55]. Setup costs are also relevant for ETO projects where the bottleneck resource may be a production department. In the following, we extend our model to allow for setup costs. We assume that for each project type to be performed, the bottleneck resource needs to be prepared (thus incurring a changeover cost). The resource remains set up for one particular project type, reflected in the *setup state*, even if the system becomes empty, until projects of another type are allocated. When that happens, a cost k^{s}_p is incurred if a project of type p is allocated while the previous project was of a different type. We introduce the extra state variable p^{s} to identify the type of the last-processed project. When allocating a project of type p', a cost $k^{\mathrm{s}}_{p'}$ is incurred if $p^{\mathrm{s}} \neq p'$. Thus, the system state is now fully chacterized by:

$$s = \left(\boldsymbol{n}^{\mathrm{W}}, \boldsymbol{n}^{\mathrm{E}}, p^{\mathrm{s}}\right). \tag{6}$$

Below, we first consider the case where the bottleneck resource cannot be idled when there are waiting projects. Afterwards, we remove this constraint and briefly discuss the benefit of allowing idleness.

5.2.1 The case without idleness

The size of the state space increases due to the extra state variable, but the number of additional states in the CTMDP can be kept limited. Further extensions such as sequence-dependent setup costs or sequence-dependent resource constraints can be easily integrated without further growth of the state space.

To investigate the benefit of flexible MPP in the presence of setup costs, we vary the setup costs k^{s}_p for the second base case of Section 4. Figure 6 shows the results, with $k^{\mathrm{s}}_1 = k^{\mathrm{s}}_2$. Overall, the average reward decreases with increasing setup costs. At the same time, the benefit of flexible MPP remains nearly stable, or slightly increases.

In order to assess the effect of setup costs on the structure of the optimal policy, we now set $k^{\mathrm{s}}_1 = k^{\mathrm{s}}_2 = 15$. First we consider the case with regular MPP. Figure 7(a) illustrates the OA decisions. Obviously, the setup state p^{s} of the bottleneck resource has a considerable

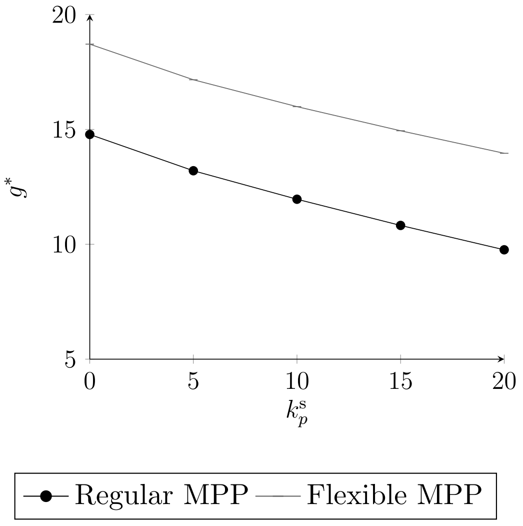
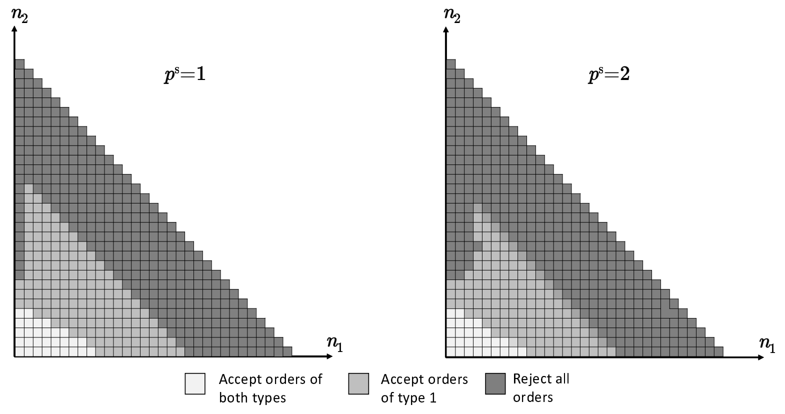


Figure 6: Benefit of flexible MPP with setup costs for the second case

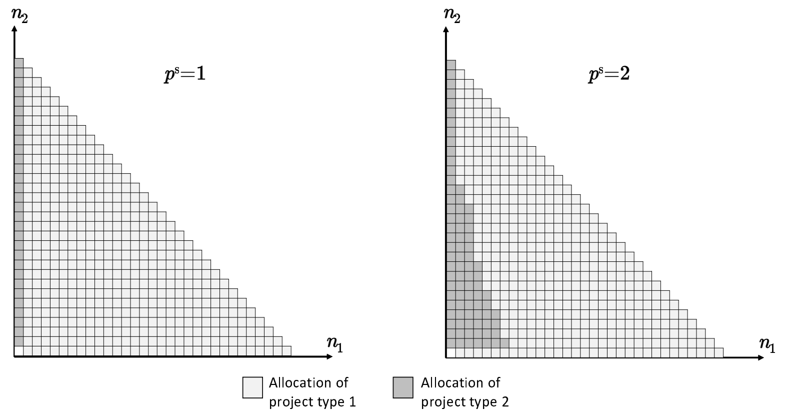
impact on the shape of the acceptance region. Furthermore, the monotone structure is less clear. If the bottleneck resource is set up for project type 1 and there are no projects of type 1 waiting or being processed, then orders of this type are rejected if there are many projects of type 2 in the system. This is logical: the system needs to be set up for type 2 at the next resource allocation decision, and this decision will be made before the arrival of the next order because preemptions are not allowed. Then, projects of type 2 are preferred and "batched" together into larger lots. If the bottleneck resource is set up for project type 2, the policy is even more selective towards orders of type 1 as a setup is always required. These observations are similar to Wester et al. [56], who find that with setup times (or setup costs) it is better to accept orders of the same type as those that are already in the system.

Figure 7(b) depicts optimal resource allocation decisions with regular MPP. If the bottleneck resource is set up for project type 1, optimal resource allocation decisions again follow a $c\mu$-policy, where projects of type 2 are only selected if no project of type 1 is in the system. When the system is set up for project type 2, however, the picture is less clear, but as long as there are only few projects of type 1 in the system, projects of type 2 are still preferred. This is in line with Hofri and Ross [22], where the project type p with the larger index $w_p\mu_p$ should be served to exhaustion. At the same time, if the system is set up for the other project type, there is a threshold on the number of projects of type p before a switch (setup) is made. Interestingly, there is also a threshold on the number of projects of type 2: if the number of projects exceeds a certain number, the system switches to type 1 (if available). This is due to the increasing holding costs for projects of type 1 waiting until completion of all type 2 projects. Beyond the threshold, it becomes better to process projects with shorter expected durations (type 1) first.

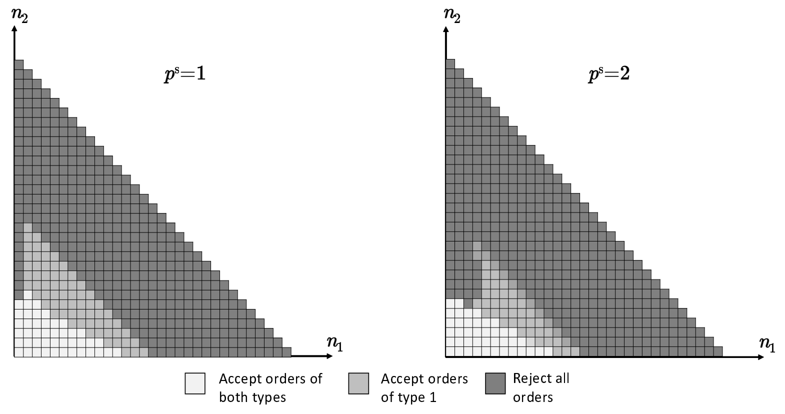
Next, we briefly analyze the effect of flexible MPP on the structure of the optimal policy with setup costs. Figure 7(c) represents the optimal OA decisions. Again, the region where both project types are accepted becomes slightly larger. The acceptance region for type 1 alone, however, is now much smaller. Its shape remains similar but it seems to "dive" into the



(a) OA decisions with regular MPP



(b) Resource allocation decisions with regular MPP



(c) OA decisions with flexible MPP

Figure 7: Results for case 2 with setup costs

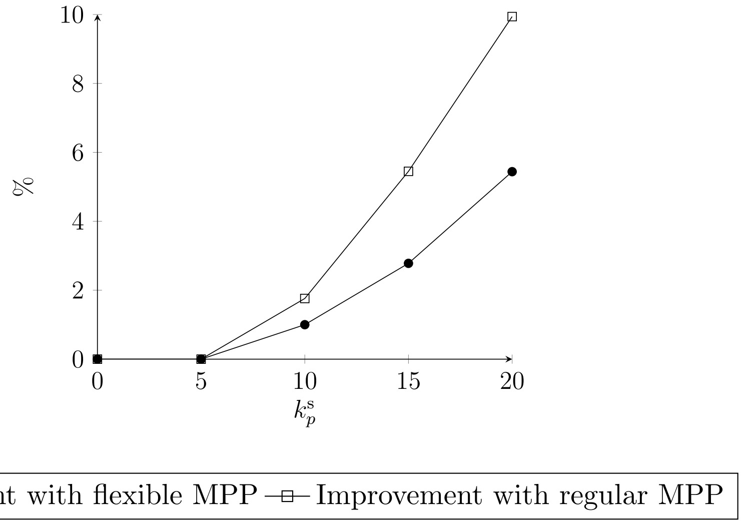


Figure 8: Benefit of idleness for the case with flexible MPP at the presence of setup costs

light grey region. The structure of the resource allocation decisions is not affected compared to regular MPP.

5.2.2 The case with idleness

Figure 8 shows the average percentage gain obtained by allowing idleness at different levels of setup costs. Only at low levels of k_p^s there is no benefit, but the difference increases quickly for values of k_p^s beyond 10. Idleness is costly, so the savings in setup costs need to be sufficiently high. The benefit of idle time is lower for flexible MPP. This can be explained by a higher objective function overall, such that the proportion of setup costs decreases, and also simply by the lower number of accepted projects (as mentioned supra).

Finally, we briefly consider the effect of idleness on the structure of the optimal policy. Figure 9 displays resource allocation decisions with regular MPP and $k_1^s = k_2^s = 15$. Compared to Figure 7(b) the structure remains very similar, but the white squares now indicate where idleness is optimal. Obviously, idleness is only optimal as long as the number of projects waiting of the other type remains below a certain threshold. This confirms the findings of Hofri and Ross [22], who (with setup times instead of costs) observe that when the processing times of both project types follow the same distribution, a double threshold policy is optimal. We see that in this case, it is not optimal to switch the setup state only if the number of projects of the current project type is zero. This is due to the different expected durations of the project types. A similar structure occurs for flexible MPP. The structure of the OA decisions is also not strongly impacted, but allowing idleness makes the policy more restrictive, in the sense that less projects are accepted, because holding cost can be reduced via idleness.

We conclude that it is beneficial to opt for idleness if setup costs are sufficiently high. The effect on the overall structure of an optimal policy is rather small, because idleness is optimal only for states where the system is near empty.

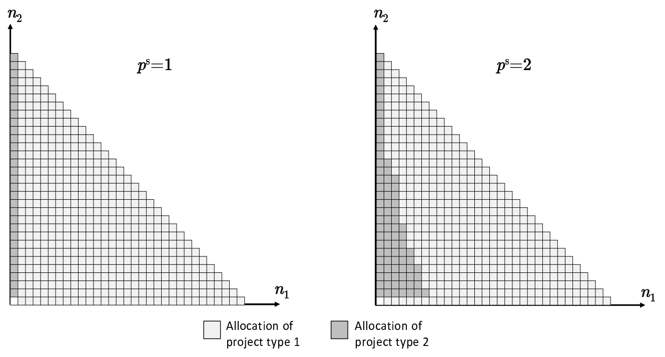


Figure 9: Optimal decisions for resource allocation for the second case with setup costs and regular MPP

6 Conclusions

In this paper, we have investigated the problem of joint order acceptance and capacity allocation on a bottleneck resource with stochastic interarrival times and stochastic project durations. Our computational study was initiated by our collaboration with an engineering department of a supplier of automotive components. In agreement with the company, we took an aggregated approach to model and analyze the problem which focused on the project work to be performed on the bottleneck machine. We have proposed a new model based on a continuous-time Markov decision process where aspects such as the option of postponing macro-process planning (MPP), usage of non-regular capacity and setup costs are taken into account.

To gain insights into the benefit of postponing MPP as well as in the structure of optimal policies, we have performed an extensive computational study. We find that it is worthwhile to postpone MPP especially when project types are very similar, when no clear dominance relationship exists and when crashing using non-regular capacity is possible. In these cases, it is less important to discriminate between project types.

In the absence of setup costs, optimal policies with respect to order acceptance and non-regular capacity have a monotone structure that makes them amenable to approximations via heuristics. In many cases, capacity allocation decisions correspond to a $c\mu$-policy. This simple structure breaks down, however, when setup costs come into play. Resource allocation decisions are strongly influenced by batching effects, and order acceptance depends on the current setup state. Thus, more sophisticated heuristics are needed for large-scale instances with higher numbers of project types. Allowing the bottleneck resource to idle is another ingredient to improve the performance when setup costs apply.

For more realistic investigations it would be advisable to focus on general distributions of interarrival times and project durations, since our models assumed the exponential distribution. Furthermore, studying projects with multiple activities and precedence constraints to be processed on multiple resources would also be a logical next step. Other directions for future research include interdependent projects, projects that arrive at certain time intervals, due dates, etc.

Appendix: Details of the Markov decision process

In this Appendix, we present further details of the reformulated continuous-time Markov decision process (CTMDP). We start with the CTMDP for the base case of Section 4 without extensions. Subsequently, we show how the extensions of Section 5 are integrated. Lastly, we address the issue of efficiently identifying an optimal policy.

6.1 CTDMP for the base problem

The CTMDP consists of *state variables*, *decision variables*, and a *transition function*. The framework has been taken from Powell [45]. The states and decision variables have been discussed supra; below we provide more information on the state transitions.

A transition from one system state s to a subsequent system state s' takes place if an event occurs after making a decision a in system state s. The time to the next event is exponentially distributed with rate:

$$\beta(s,a) = \sum_{p\in\mathcal{P}} n_p^{\mathrm{E}}(s)\mu_p + \sum_{p\in\mathcal{P}} \lambda_p. \tag{7}$$

Next, we formally state the mappings to the subsequent states upon an arrival or completion event. In what follows, we use $\boldsymbol{e}_p$ for a unit vector of dimension $|\mathcal{P}|$ having a value of 1 at position p and zero for all other positions.

1. Arrival of an order. On arrival of a new order of type $p \in \mathcal{P}$ that is accepted, the subsequent system state is given by:

$$s' = \begin{cases} \left(\boldsymbol{n}^{\mathrm{W}}(s) + \boldsymbol{e}_p\boldsymbol{\delta}^{\mathrm{AM}}, \boldsymbol{n}^{\mathrm{E}}(s)\right) & \text{if } s \neq s^0, \\ (\boldsymbol{0}, \boldsymbol{e}_p) & \text{if } s = s^0, \end{cases} \tag{8}$$

   where $s^0 = ((0,\ldots,0),(0,\ldots,0))$ represents an empty system. On arrival, as long as there is a project in process ($s \neq s^0$), an accepted order of type p becomes a project and is added to the waiting projects. If there is no project in process at system state s^0, the order is immediately allocated to the resource.

2. Completion of a project. The subsequent state is given by:

$$s' = \begin{cases} \left(\boldsymbol{n}^{\mathrm{W}}(s) - \boldsymbol{e}_p\boldsymbol{\delta}^{\mathrm{E}}, \boldsymbol{\delta}^{\mathrm{E}}\right) & \text{if } s \neq s^0, \\ s^0 & \text{if } \boldsymbol{n}^{\mathrm{W}}(s) = \boldsymbol{0}. \end{cases} \tag{9}$$

   In this case, the project in process is removed from the system and the project given by $\boldsymbol{\delta}^{\mathrm{E}}$ is allocated. At the end of the transition a fixed payoff y_p is obtained.

6.2 Extensions of the CTMDP

For the inclusion of non-regular capacity, the only modification in addition to those outlined in Section 5.1 is that the rate to the next event is now given by:

$$\beta(s,a) = \sum_{p\in\mathcal{P}} n_p^{\mathrm{E}}(s)\mu_p(1 + z_p \cdot \delta^{\mathrm{C}}) + \sum_{p\in\mathcal{P}} \lambda_p. \tag{10}$$

With respect to setup costs (Section 5.2), a number of changes need to be incorporated. First, we adopt a new state definition in order to keep track of the setup state of the system. The state of the system is defined as a triple $s = (\boldsymbol{n}^W, \boldsymbol{n}^E, p)$, where p denotes the type of the project that is currently being processed, or (in case the system is empty/idle) was last processed. Let $\mathcal{S}^{\mathrm{E}}$ denote the statespace of the CTMDP with setups (note that $\mathcal{S}^{\mathrm{E}}$ is larger than $\mathcal{S}$). Secondly, we need to take into account setup costs when changing the project type upon dispatching a project. If the system is in setup state p, a setup cost $k^s_{p'}$ is incurred if the dispatched project is of type $p' \neq p$.

6.3 Solution methodology

We address how the structure of the problem can be exploited in order to efficiently determine an optimal policy. We first establish some fundamental properties.

Theorem 1. *Under any stationary policy π, performing OA and capacity planning decisions in $\mathcal{S}$, the CTMDP with statespace $\mathcal{S}$ is unichain.*

Proof. It is sufficient to restrict the consideration to the resource allocation decisions made by a policy. To start we note that, by restricting to non-idling policies, the resource is always busy after a decision, except if the system is empty: the resource is busy for all states $s \in \mathcal{S}\backslash\{s^0\}$. Furthermore, with non-zero probability, no new project arrives or is accepted until the system becomes empty, such that system state s^0 is accessible from any system state $s \in \mathcal{S}\backslash\{s^0\}$. Conversely, under a policy π, a subset $\mathcal{S}(\pi) \subseteq \mathcal{S}\backslash\{s^0\}$ is accessible from s^0 such that all $s \in \mathcal{S}(\pi)$ communicate with s^0 and $\mathcal{S}(\pi) \cup \{s^0\}$ is a recurrent class of states. Since s^0 is accessible from all transient states $s \in \mathcal{S}\backslash\mathcal{S}(\pi)$, it is the only recurrent class. $\square$

For the CTDMP with state space $\mathcal{S}$ the above result implies that, for any stationary policy π, there exists a single $g(\pi)$ independent from the starting state. Thus, we can apply *policy iteration* with *value iteration* for policy evaluation in order to determine an optimal policy. For details on these methods, we refer to Puterman [46].

Theorem 2. *The CTMDP with state space $\mathcal{S}^{\mathrm{E}}$ is multichain and weakly communicating.*

Proof. We first show that the CTMDP with $\mathcal{S}^{\mathrm{E}}$ is multichain. We construct a policy with the following property. As soon as the system is in one of the empty states $s^0_p = (\mathbf{0}, \mathbf{0}, p)$, only orders of type p are accepted afterwards. Obviously, the system never changes the setup state anymore and may return only to s^0_p with non-zero probability. As a consequence, we obtain $|\mathcal{P}|$ recurrent classes of states.

To prove that the CTMDP is weakly communicating, we construct a policy that accepts orders of any type until the maximum number of projects in the system has been reached. Since the policy accepts projects of all types, any state $s \in \mathcal{S}^{\mathrm{E}}$ can be accessed as long as $\lambda_p > 0$ for all $p \in \mathcal{P}$. To see this, we construct a sample path that starts from state s^0_p. Assume that a project of type p arrives first (thus it must be immediately allocated to the resource). Afterwards, arriving projects of each type $p \in \mathcal{P}$ are waiting while the first project is being processed. It is clear that each state $s^0_{p'}$ with $\lambda_{p'} > 0$ must be accessible from s^0_p as there exists a sample path with non-zero probability where only orders of type p' arrive until the system becomes empty again (ending up in $s^0_{p'}$). Clearly, s^0_p is accessible from $s^0_{p'}$ if

$\lambda_p > 0$. Thus, the states s_p^0 for all $p \in \mathcal{P}$ with $\lambda_p > 0$ are communicating and there exists a single recurrent class of states under such a policy. $\square$

The multichain property implies that there exist policies where $g(\pi)$ depends on the starting state of a sample path. As the CTMDP is also weakly communicating, however, it can be shown that there exists an optimal policy π^* with a single $g(\pi^*)$, which is independent from the starting state (cf. Puterman [46]). This implies that intermediate policies in policy iteration may have multiple $g(\pi)$, whereas the optimal policy has a single $g(\pi^*)$. One option in this case would be to apply an adaptation of policy iteration where intermediate policies π are modified to obtain a single $g(\pi)$ [46]. We have found that it is sufficient for our problem to abort value iteration after a given number of iterations instead of iterating until convergence to a single value.

References

[1] P. S. Adler, A. Mandelbaum, V. Nguyen, and E. Schwerer. From project to process management: An empirically-based framework for analyzing product development time. *Management Science*, 41(3):458–484, 1995.

[2] S. Anavi-Isakow and B. Golany. Managing multi-project environments through constant work-in-process. *International Journal of Project Management*, 21:9–18, 2003.

[3] J. Bard, R. Balachandra, and P. Kaufman. An interactive approach to R&D project selection and termination. *IEEE Transactions on Engineering Management*, 35:139–146, 1988.

[4] E.B. Berman. Resource allocation in a PERT network under continuous activity time-cost functions. *Management Science*, 10(4):734–745, 1964.

[5] J.W.M. Bertrand and D.R. Muntslag. Production control in engineer-to-order firms. *International Journal of Production Economics*, 30-31:3–22, 1993.

[6] N. Brahimi, T. Aouam, and E.H. Aghezzaf. Integrating order acceptance decisions with flexible due dates in a production planning model with load-dependent lead times. *International Journal of Production Research*, 53(12):3810–3822, 2015.

[7] I. Cohen, V. Nguyen, and A. Shtub. Multi-project scheduling and control: A process-based comparative study of the Critical Chain methodology and some alternatives. *Project Management Journal*, 35(2):39–50, 2004.

[8] D.R. Cox and W.L. Smith. *Queues*. Methuen, 1961.

[9] T.B. Crabill. Optimal control of a service facility with variable exponential service times and constant arrival rate. *Management Science*, 18(9):560–566, 1972.

[10] R. de Boer. *Resource-Constrained Multi-Project Management*. PhD thesis, Universiteit Twente, The Netherlands, 1998.

[11] Y. De Serres. Simultaneous optimization of flow control and scheduling in a single server queue with two job classes. *Operations Research Letters*, 10:103–112, 1991.

[12] Y. De Serres. Simultaneous optimization of flow control and scheduling in a single server queue with two job classes: Numerical results and approximation. *Computers and Operations Research*, 18:361–378, 1991.

[13] Y. De Serres. *Simultaneous optimization of flow control and scheduling in queues.* PhD thesis, McGill University, Montreal, Canada, 1991.

[14] F.F. Easton and D.F. Rossin. Overtime schedules for full-time service workers. *Omega*, 25:285–299, 1997.

[15] M.J.R. Ebben, E.W. Hans, and F.M. Olde Weghuis. Workload based order acceptance in job shop environments. *OR Spectrum*, 27:107–122, 2005.

[16] E. A. Feinberg and F. Yang. Optimality of trunk reservation for an M/M/k/N queue with several customer types and holding costs. *Probability in the Engineering and Informational Sciences*, 25(4):537–560, 2011.

[17] J. C. Gittins and D. M. Jones. A dynamic allocation index for the sequential design of experiments. In János Bolyai, editor, *Progress in statistics (European Meeting of Statisticians, Budapest)*, volume 9. Colloquium Mathematical Society, 1972.

[18] E.M. Goldratt. *Critical Chain.* The North River Press, 1997.

[19] E.W. Hans. *Resource Loading by Branch-and-Price Techniques.* PhD thesis, University of Twente, The Netherlands, 2001.

[20] E.W. Hans, W. Herroelen, R. Leus, and G. Wullink. A hierarchical approach to multi-project scheduling under uncertainty. *Omega*, 35(5):563–577, 2007.

[21] J. Herbots, W. Herroelen, and R. Leus. Dynamic order acceptance and capacitiy planning on a single bottleneck resource. *Naval Research Logistics*, 54:874–889, 2007.

[22] M. Hofri and K.W. Ross. On the optimal control of two queues with server setup times and its analysis. *SIAM Journal on Computing*, 16(2):399–420, 1987.

[23] T. Hutter, S. Haeussler, and H. Missbauer. Successful implementation of an order release mechanism based on workload control: A case study of a make-to-stock manufacturer. *International Journal of Production Research*, to appear.

[24] C.V. Ivanescu, J.C. Fransoo, and J.W.M. Bertrand. Makespan estimation and order acceptance in batch process industries when processing times are uncertain. *OR Spectrum*, 24:467–495, 2002.

[25] V.C. Ivanescu, J.C. Fransoo, and J.W.M. Bertrand. A hybrid policy for order acceptance in batch process industries. *OR Spectrum*, 28:199–222, 2006.

[26] S. Kavadias and C.H. Loch. Optimal project sequencing with recourse at a scarce resource. *Production and Operations Management*, 12(4):433–442, 2003.

[27] J.E. Kelley and M.R. Walker. Critical-path planning and scheduling. In *1959 Proceedings of the Eastern Joint Computer Conference*, pages 160–173, 1959.

[28] A.J. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35, 1998.

[29] A.J. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41, 2001.

[30] G.P. Klimov. Time-sharing service systems I. *Theory of Probability and its Applications*, 19(3):532–551, 1974.

[31] N.C. Knudsen. Individual and social optimization in a multiserver queue with a general cost-benefit. *Econometrica*, 40(3):515–528, 1972.

[32] N. Levy and S. Globerson. Improving multiproject management by using a queueing theory approach. *Project Management Journal*, 28(4):40–46, 1997.

[33] Z. Liu, P. Nain, and D. Towsley. On optimal polling policies. *Queueing Systems*, 11:59–83, 1992.

[34] C.H. Loch and S. Kavadias. Dynamic portfolio selection of NPD programs using marginal returns. *Management Science*, 48(10):1227–1241, 2002.

[35] C.H. Loch, M.T. Pich, M. Urbschat, and C. Terwiesch. Selecting R&D projects at BMW: A case study of adopting mathematical programming methods. *IEEE Transactions on Engineering Management*, 48(1):70–80, 2001.

[36] M. I. McManus. Optimum use of overtime in post offices. *Computers and Operations Research*, 4:271–78, 1977.

[37] P. Melchiors. *Dynamic and Stochastic Multi-Project Planning*, volume 673 of *Lecture Notes in Economics and Mathematical Systems*. Springer International Publishing, 2015.

[38] P. Melchiors and R. Kolisch. Scheduling of multiple R&D-projects in a dynamic and stochastic environment. In B. Fleischmann, K.H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008*, pages 135–140. Springer Berlin Heidelberg, 2009.

[39] P. Melchiors and R. Kolisch. Markov-decision-process-based scheduling of multiple projects in a stochastic dynamic environment. In *Proceedings of the 12th International Conference on Project Scheduling and Management*, pages 295–298, 2010.

[40] S. Meyn. *Control Techniques For Complex Networks*. Cambridge University Press, 2008.

[41] P. Naor. The regulation of queue size by levying tolls. *Econometrica*, 37(1):15–24, 1969.

[42] K. Neumann, C. Schwindt, and J. Zimmermann. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions.* Springer Science & Business Media, 2012.

[43] J. Niño-Mora. Stochastic scheduling. In P.M. Pardalos, editor, *Encyclopedia of Optimization*, volume V, pages 367–372. Kluwer Academic Publishers, 2005.

[44] T.C. Perry and J.C. Hartman. Allocating manufacturing capacity by solving a dynamic, stochastic multiknapsack problem. Technical Report ISE 04T-009, Lehigh University, Pennsylvania, 2004.

[45] W.B. Powell. *Approximate Dynamic Programming.* John Wiley and Sons, Inc., Hoboken, New Jersey, 2nd edition, 2011.

[46] M.L. Puterman. *Markov Decision Processes – Discrete Stochastic Dynamic Programming.* Wiley, 1994.

[47] M.I. Reiman and L.M. Wein. Dynamic scheduling of a two-class queue with setups. *Operations Research*, 46(4):532–547, 1998.

[48] J. Riezebos. Order sequencing and capacity balancing in synchronous manufacturing. *International Journal of Production Research*, 49(2):531–552, 2011.

[49] T.A. Roemer and R. Ahmadi. Concurrent crashing and overlapping in product development. *Operations Research*, 52(4):606–622, 2004.

[50] K.W. Ross and D.H.K. Tsang. Optimal circuit access policies in an ISDN environment: A Markov decision approach. *IEEE Transactions on Communications*, 37(9):934–939, 1989.

[51] R.L. Schmidt and J.R. Freeland. Recent progress in modeling R&D project-selection processes. *IEEE Transactions on Engineering Management*, 39(2):189–201, 1992.

[52] S. Slotnick and T. Morton. Order acceptance with weighted tardiness. *Computers and Operations Research*, 34:3029–3042, 2007.

[53] S. Stidham and R. Weber. A survey of Markov decision models for control of networks of queues. *Queueing Systems*, 13:291–314, 1993.

[54] F. Talla Nobibon and R. Leus. Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment. *Computers and Operations Research*, 38(1):367–378, 2011.

[55] N.D. van Foreest, J. Wijngaard, and J.T. van der Vaart. Scheduling and order acceptance for the customized stochastic lot scheduling problem. *International Journal of Production Research*, 48(12):3561–3578, 2010.

[56] F.A.W. Wester, J. Wijngaard, and W.H.M. Zijm. Order acceptance strategies in a production-to-order environment with setup time and due-dates. *International Journal of Production Research*, 30(6):1313–1326, 1992.

[57] P. Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25:287–298, 1988.

[58] U. Yechiali. On optimal balking rules and toll charges in the GI/M/1 queueing process. *Operations Research*, 19:349–370, 1969.

[59] U. Yechiali. Customers' optimal joining rules for the GI/M/s queue. *Management Science*, 18:434–443, 1972.

[60] W.H.M. Zijm. Towards intelligent manufacturing planning and control systems. *OR Spectrum*, 22:313–345, 2000.