universität wien

Magisterarbeit

Titel der Magisterarbeit

"Using simulation optimization and genetic algorithms to solve single-machine, multiple-item scheduling problems with sequence-dependent setup times and stochastic demands"

Verfasser

Manuel Riel

angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften

(Mag. rer. soc. oec.)

Wien, im November 2011

Studienkennzahl lt. Studienblatt:	A 066 914
Studienrichtung lt. Studienblatt:	Magisterstudium Internationale Betriebswirtschaft
Betreuer:	Univ.-Prof. Dr. Stefan Minner

Abstract

This text deals with the stochastic version of the Economic Lot Sizing Problem (ELSP) with sequence-dependent setup times. The problem arises from the fact that it is often more profitable to invest in a single high-capacity machine that can be used for multiple purposes, than cheaper specialized machines. Deterministic and setup-independent versions of the problem have been researched extensively. Research approaches that combine both are relatively sparse.

The paper combines some earlier works, as well as metaheuristic concepts to set better production policies. A genetic algorithm is used to improve the sequence and minimize setup times of the whole cycle. Then a global optimizer evaluates policies in simulation runs and tries to improve the result by adapting base stock levels and product frequencies. The results are evaluated in a numerical study.

Zusammenfassung

Diese Arbeit behandelt die stochastische Version des Economic Lot Sizing Problem (ELSP) mit sequenzabhängigen Setupzeiten. Das Problem entwickelt sich aus der Tatsache, dass es in vielen Fällen günstiger ist, eine teurere Maschine mit höherer Kapazität und breiteren Einsatzmöglichkeiten anzuschaffen, als eine spezialisierte Maschine, die nur für einen spezifischen Vorgang einzusetzen ist. Die deterministische Version des Problems wird seit einigen Jahrzehnten behandelt und es ist eine große Masse an Literatur dazu verfügbar. Auch die sequenzabhängige Version des deterministischen ELSP-Problems ist als Abwandlung des Traveling Salesman Problem (TSP) relativ gut erforscht. Arbeiten, die beide Probleme kombinieren sind hingegen fast nicht vorhanden.

Aus diesem Grund versucht die vorliegende Arbeit, Probleme mit beiden Einschränkungen simultan zu lösen. Dazu werden vorhandene Arbeiten aus beiden Bereichen ausgewertet und relevante Teile kombiniert. Metaheuristiken aus dem Gebiet des deterministischen ELSP werden dazu verwendet, um Startparameter für Simulationsoptimierung zu erhalten. Anschließend versucht ein globaler Optimierer die Lösung sukzessive zu verbessern. Zum Setzen der Produktionssequenzen – welche bei sequenzabhängigen Setupzeiten entscheidend sind – wird ein genetischer Algorithmus aus dem Bereich der TSP-Forschung verwendet.

Im numerischen Teil der Arbeit werden fünf verschiedene Produktionsplanungsregeln verglichen. Zwei davon basieren auf einem gemeinsamen Zyklus, der für alle Produkte gleich sind. Die restlichen drei Regeln implementieren fixe Zyklusfolgen, in denen ausgewählte Produkte mehrfach vorkommen können. Die letztgenannten Regeln werden in Hinsicht auf das Optimierungsziel verglichen. Der Zyklus wird entweder auf Ausgeglichenheit, Kosteneffizienz in Hinsicht auf Setupzeiten oder eine Kombination aus den beiden optimiert.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	Ant Colony.
ADP	Approximate Dynamic Programming.
AI	Artificial Intelligence.
ANN	Artificial Neural Network.
B&B	Branch and Bound.
BP	Basic Period.
CC	Common Cycle.
CG	Common cycle with genetic sequence.
CMA-ES	Covariance Matrix Adaption Evolutionary Strategy.
CN	Common cycle with closest neighbor sequence.
DIY	do-it-yourself.
DNA	Deoxyribonucleic acid.
DP	Dynamic Programming.
EBP	Extended Basic Period.
ELSP	Economic Lot Sizing Problem.
EOQ	Economic Order Quantity.
FC	Fixed Cycle.
FGI	Finished Goods Inventory.
FH	Fixed cycle policy with minimized holding cost.
FM	Fixed cycle with multi-objective sequence.
FS	Fixed cycle policy with minimized setup cost.
GA	Genetic Algorithm.
GAMS	General Algebraic Modeling System.
GRASP	Greedy Randomized Adaptive Search Procedure.
INLP	Integer Non-Linear Programming.
IP	Internet Protocol.
LMS	Least Mean Squares.
LP	Linear Programming.
MAD	Mean Average Deviation.
MDP	Markov Decision Process.
MRP	Material Requirements Planning.

NN	Nearest/Closest Neighbor.
RCL	Restricted Candidate List.
RL	Reinforcement Learning.
RO	Runout Time.
SA	Simulated Annealing.
SD-ELSP	Sequence-Dependent ELSP.
SD-SELSP	Sequence-Dependent Stochastic ELSP.
SDS	Sequence-Dependent Setup times.
SELSP	Stochastic Economic Lot Sizing Problem.
SIS	Sequence-Independent Setup times.
SKU	Stock-Keeping Unit.
SMDP	Semi-Markov Decision Process.
TS	Tabu Search.
TSP	Traveling Salesman Problem.
VM	Virtual Machine.
WBGA	Weight-based Genetic Algorithm.

1 Introduction

This work deals with the stochastic version of the Economic Lot Sizing Problem (ELSP) with Sequence-Dependent Setup times (SDS). The different dimensions of the problem are evaluated and solved by using a number of optimization methods to find an optimal policy.

1.1 Problem Introduction

The ELSP is one of the oldest challenges of production planning and has been around since the first production plants were built in the early 19th century (Maxwell, 1964, p. 89). The problem arises from simple economics. In many cases it's cheaper to invest in a single high-capacity machine that can be used for multiple purposes.

The machine can only produce one item at a time, so items need to be scheduled in a way that optimizes the production manager's objectives. These are commonly the reduction of production costs, i.e. inventory cost, setup cost and the implicit cost of lost sales or backlogs. Actual expenditures for production or fixed cost positions are not taken into account, because they are hard to influence by only using improved production schedules.

When research in ELSP started, many tools, like global optimizers and simulation models were not available. Moreover factories tended to produce a smaller range of products in big quantities to gain economies of scale and provide cheap consumer goods to high-growth markets. Supply chains were longer with a series of wholesalers and importers between producer and consumer. This covered the true cost of holding excess inventory, or even required more of it to account for long lead times.

With so many intermediaries, the manufacturer had little information about actual market demand. He would only learn about it through proxies, who often distorted the picture by batch-ordering or seasonal sell-offs.

In such an environment of high inventory and with little information available, production decisions were mostly based on experience and anticipated orders, rather than actual sales forecasts. Production managers required models, allowing them to efficiently schedule orders they already knew about, rather than expected ones.

Under these conditions, the first ELSP models were constructed. They made many unrealistic assumptions and were mostly based on the popular Economic Order Quantity (EOQ) model, with a focus on finding the right batch size at a constant demand rate.

When better communication technologies and cheap information processing became widely available, companies started collection data about their markets and analyzed it. What they found were patterns of seasonality and probability. While precise predictions were still difficult, demand could be modeled within probability distributions.

At the same time computer simulation allowed for complex production policies could be tested in a virtual environment, before unleashing them at the real world. The difficulty of proving the superiority of innovative policies became less dependent on the risk-appetite of the factory manager. This should lead to a natural increase in innovation.

The same forces that made companies more knowledgable, also brought them new challenges. Better market information empowered consumers and gave rise to new competitors from regions as remote as China and Japan. This meant better prices for consumers, but also lower margins for firms. Suddenly excess inventory and misguided production decisions made a substantial difference in bottom line. Every surplus piece of inventory decreased a firm's contribution margin.

All of these influences can be seen in more recent ELSP models that do away with some wishful model assumptions and make production models more robust and useful. The pure cyclical view of finding the perfect cycle and sticking to it was replaced by dynamic control policies, accounting for an equally dynamic and ever-changing environment.

Stochastic features and dynamic policies were an important step for some practitioners. For others, it was more important to remove another unrealistic assumption: sequence-independent setup costs. Even though setup costs were often considered, most models assumed them only to be tied to the product *currently* being produced. While this was true for some environments, studies (Allahverdi et al., 1999, p. 219) have shown that as many as three quarters of production managers report SDS for at least *some* of their processes. This class of setup costs is especially relevant in chemical production processes, or when dealing with families of products. Setup costs might be low when switching *within* a family, but higher when switching *between* families.

When taking all of the above mentioned trends and requirements into account, it is surprising that almost no researchers took the step to relax both restrictions at the same time, namely sequence-dependent setup costs and stochastically distributed demand. By combining earlier research and tools that relax one or the other condition, it should be possible to adequately solve such a model and improve the overall outcome.

1.2 Practical Applications

Chapter 3 describes two supply chains that would benefit from a solid model to solve Sequence-Dependent Stochastic ELSP (SD-SELSP). The first case study deals with a pharmaceutical com-

pany. Products need to be packaged for sale in three different European countries. The case-study proposes centralized demand- and production planning, as well as one centralized inventory instead of three local ones. In such a situation the deterministic production scheduling problem that includes up to three weeks of collected orders becomes a cyclical schedule with stochastic demands.

The second practical example portrays a manufacturer of pre-Deco material. Inventory is kept at eight different locations and production planning happens ad-hoc. This leads to excess inventory at the distributors and high production costs due to unnecessary switching between product families. An optimized production sequence with the right frequencies should help the company to decrease production- and inventory cost.

1.3 Outline of the paper

The work starts with an overview of existing literature and solution approaches of the problem, which can be used to solve the SD-SELSP. The succeeding chapter gives a description of the problem instances that need to be solved to arrive at a complete production schedule. Then an illustration of the model, its assumptions and control policies follows. The subsequent chapter introduces the theoretical foundations of the solution methods that were applied to the problem. In chapter number six the model is tested on a series of numerical problem instances and conclusions on the performance of proposed control policies are drawn. The final chapter gives a conclusion, as well as possibilities for further research.

2 Literature review

This chapter gives an overview of existing work related to the Stochastic Economic Lot Sizing Problem (SELSP) with SDS. Being one of the most fundamental problems in production planning, this topic has been researched extensively. Especially for the closely related deterministic ELSP, there is a large number of heuristics and analysis available. Even though this body of research is not directly applicable to the stochastic version of the problem, it can be a useful source of heuristics.

The SDS-version of the problem also exists in both, the stochastic and deterministic domain. It removes an essential relaxation of the original model, in that setup times are now dependent on the previously manufactured product. This brings the model closer to practical applications.

2.1 Deterministic ELSP

Ever since the introduction of industrial production facilities at the beginning of the 20th century, factory managers had a motivation to use capital-intensive machines as efficiently as possible. Using a single production facility to produce more than one kind of product is a good way to spread the capital expenditure and risk over a number of products.

With such a setup, one will quickly face conflicts between multiple products competing for a spot on a single machine with finite capacity.

To resolve such conflicts, while at the same time minimizing setup- and holding costs, a number of approaches have been developed. Depending on the quality of the result and the relationship to the original problem, they can be classified in *analytical* and *heuristic* approaches (Elmaghraby, 1978, p. 587).

Analytical methods solve a simplified version of the situation, but deliver the optimal solution for it. They are a good choice if the original problem can be abstracted into the required model without much loss of relevance.

Heuristic approaches provide a solution to the native problem, but don't guarantee that it is the optimal one. Depending on the size of the gap between prime- and heuristic solution, reduced processing time can be worth the tradeoff.

Even though the analytical- and heuristic approaches reviewed in Elmaghraby (1978) can't be used to directly solve SELSP with SDS, they provide the basic solution framework for more

sophisticated solutions and their heuristics can be used to speed up global search heuristics. For that reason the most popular approaches will still be reviewed here.

2.1.1 Common (rotation) cycle solution

The Common Cycle (CC) is the simplest solution algorithm and also relatively easy to calculate. It always produces a feasible solution, which might not be optimal under all conditions (A. Raza & Akgunduz, 2008, p. 95).

This simplicity comes at the cost of a strict model, requiring an unrealistic set of assumptions that are stated in Maxwell (1964) and conveniently summarized in A. Raza and Akgunduz (2008, p. 94):

1. The machine can only produce one product at a time.
2. Inventory holding cost is directly proportional to the amount of inventory.
3. The sum of production- and setup times does not exceed the total capacity of the production facility.
4. Setup cost and times are independent of the production sequence.
5. Each product has deterministic and constant demand- and production rates.

The first limitation ensures that no dependencies between products exist and no two products are manufactured in parallel. There might exist production processes that feature parallel processing or side products (as in the chemical industry), but they are too specific to be taken into account in a generalized model.

The second assumption will also not hold in all situations, as conventional warehouse space can not be increased in a linear fashion. But as for the first point, any other assumption would harm general applicability.

Assumption number three makes sure that the production facility never runs at a capacity of $> 100\%$. This is important because at such high capacity utilization, the optimal solution would include a tradeoff between different products that can not be produced. Such a decision would not be possible with the information traditionally found in a production planning problem. It would also require some data about sales price and contribution margin.

The two final assumptions can also be found in a more recent classification of SELSP in Winands et al. (2011, p. 2). They will later be relaxed for a more realistic and versatile model.

Notations and basic relations

This paper will use the notations found in Elmaghraby (1978), where applicable:

i	item index
d_i	demand rate in units per time unit
p_i	production rate in units per time unit
s_i	setup time in units of time per production lot; still independent of sequence
A_i	setup cost per production run; independent of sequence
h_i	holding cost per product unit per unit of time
T_i^*	(optimal) cycle time for product i

With these values, the cost for an individual product i, when assuming a certain cycle length T_i can be calculated (Elmaghraby, 1978, p. 588):

$$C_i = \frac{A_i}{T_i} + h_i d_i \frac{T}{2}(1 - \frac{d_i}{p_i}) \tag{2.1.1}$$

By differentiating this expression, the cost-effective cycle length for an individual product can be found. This formula is actually quite close to the well-known EOQ formula introduced by Harris (1913).

$$T^* = \sqrt{\frac{2A_i}{h_i d_i (1 - \frac{d_i}{p_i})}} \tag{2.1.2}$$

By summing over these expressions, Maxwell (1964, p. 92) calculates the total cost for all products in the cycle.

$$C = \sum_{i=1}^{n} \frac{A_i}{T} + \frac{T}{2} \sum_{i=1}^{n} h_i d_i (1 - \frac{d_i}{p_i}) \tag{2.1.3}$$

The optimal T^* can then be found in a similar way as (2.1.2):

$$T^* = \sqrt{\frac{2 \sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} h_i d_i (1 - \frac{d_i}{p_i})}} \tag{2.1.4}$$

Since the total cycle time is restrained by the time required for setup, the lower bound for T^* is

$$T^* \geq \frac{\sum_{i=1}^{n} s_i}{1 - \frac{d_i}{p_i}} \tag{2.1.5}$$

So the complete solution is

$$T^* = \max\left(\sqrt{\frac{2\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} h_i d_i (1 - \frac{d_i}{p_i})}}, \frac{\sum_{i=1}^{n} s_i}{1 - \frac{d_i}{p_i}}\right) \tag{2.1.6}$$

As can be seen from formulas 2.1.1 to 2.1.6, the CC-approach features a simple and convenient analytical approach to solve basic ELSP-problems. Unfortunately simplicity comes at a price and there are a number of disadvantages when using CC. Maxwell (1964, p.92-93) summarizes arguments against using a rotation cycle:

First of all, it only works well, when there are no gross imbalances in demand rates, production rates and setup times (or SDS). In practice this is not very often the case, as demand is often concentrated on a few items that cause most of the volume. This effect is often referred to as Pareto 80/20-rule or ABC-analysis (Swamidass, 2000). When high-volume products don't add too much to (sequence dependent) setup times, producing them multiple times in a cycle can lower costs significantly.

2.1.2 Basic Period

Due to the sub-optimalities observed with CC, researchers came up with a more flexible approach that was designed to solve some of the problems caused by a rotation cycle. Bomberger (1966) suggests a Basic Period (BP) (also called Fixed Cycle (FC)) approach, extending CC by an integer multiplier k_i. With this approach, high-volume items are allowed to have their own production cycle, based on a common BP T. Before extending the CC-formulas to account for different cycle lengths, the following new notations need to be introduced:

k_i integer multiple describing how often an item is produced within a full cycle
T basic period (sometimes referred to as W)

First the relationship between k_i, T and T_i

$$T_i = k_i T \tag{2.1.7}$$

on the condition that

$$\sum_{i=1}^{n} k_i T \frac{d_i}{p_i} \leq T \tag{2.1.8}$$

By substituting T with (2.1.7) in (2.1.1), as suggested in (Elmaghraby, 1978, p. 781), the following new target function can be deduced:

$$\min C(k_i, T) = \sum_{i=1}^{n} \frac{A_i}{k_i T} + \frac{k_i T}{2} \sum_{i=1}^{n} h_i d_i (1 - \frac{d_i}{p_i}) \tag{2.1.9}$$

Having defined a target function and constraints, the resulting problem is NP-hard and can not be solved to optimality directly. Even a restricted version of the problem which limits integer multiples to the power of 2 is still NP-hard (Hsu, 1983).

In his original paper, Bomberger (1966, p. 782) suggests a Dynamic Programming (DP) solution to find optimal parameters. This is computationally expensive and requires a large number of DP-problems to be solved. Therefore substantial efforts have been made to solve the problem heuristically or metaheuristically. One is a Genetic Algorithm (GA) approach by Khouja et al. (1998). They were able to improve the DP solution of the problem by 2.15%, even for problems with up to 30 products. A recent comparison of different heuristics can also be found in A. Raza and Akgunduz (2008).

A well-known heuristic for solving the BP approach was popularized by Doll and Whybark (1973). Though their solution has been outperformed (slightly) by computationally more elaborate solutions, the heuristic gives good solutions for the BP problem and is also quite easy to understand and implement.

They criticize that existing solutions of the basic period ELSP rely on human judgement to determine T_i. This value is usually judged on the basis of comparing individual T_i. As mentioned in the initial part of this chapter, the starting point for the estimate is most commonly EOQ.

Doll and Whybark (1973) strive to determine T_i directly in an iterative approach:

Initially T_i^* for each product is determined by using (2.1.2).

Next the smallest T_i is used for T.

$$T = \min[T_i^*] \tag{2.1.10}$$

Subsequently integer multiples k_i^- and k_i^+ for each product are selected.

k_i^- next lowest integer multiple
k_i^+ next highest integer multiple

$$k_i^- \leq \frac{T_i^*}{T} \leq k_i^+ \tag{2.1.11}$$

By using (2.1.3) and (2.1.7), the cost for a certain integer decision is given by

$$C_i = \frac{A_i}{k_i T} + \frac{h_i d_i k_i T}{2}(1 - \frac{d_i}{p_i}) \tag{2.1.12}$$

Once new k_i are chosen by finding the lowest cost integer, an aggregate version of (2.1.12) can be used to calculate total cost of a given basic period T.

$$C(T) = \sum_{i=1}^{n} \frac{A_i}{k_i T} + \frac{T}{2} \sum_{i=1}^{n} h_i d_i k_i (1 - \frac{d_i}{p_i}) \tag{2.1.13}$$

By differentiating, the cost-minimizing T can be found

$$T = \sqrt{\frac{2 \sum_{i=1}^{n} \frac{A_i}{k_i}}{\sum_{i=n}^{n} h_i d_i k_i (1 - \frac{d_i}{p_i})}} \tag{2.1.14}$$

After determining a new T, the procedure returns to choosing new k_i^+ and k_i^+. Once two runs produce identical values of k_i, the algorithm stops.

Doll and Whybark (1973) conclude that their heuristic is an effective alternative to solving analytical ELSP, such as Bomberger (1966), but also acknowledge that there are situations where the procedure can not be used directly. Elmaghraby (1978, p. 593) points out two other problems the algorithm might run into. First, it gives no guidance when the solution parameters produced are infeasible. In such a case, a practitioner would need to resort to manually adjusting k_i to achieve a feasible solution. Second, if multipliers are then adjusted in such a manual way, the stopping criteria may never be reached.

One major limitation of the basic period approach is the need for equal lot sizes during production runs. This was first realized by Elmaghraby (1978), who suggested using two cycles of length T at a time and loading products by distinguishing between even and odd n_k. He called the method Extended Basic Period (EBP).

While this new EBP-approach brings some improvement over the best possible plain basic period model and the recent application of a GA solves it to near-perfection, the approach itself doesn't provide enough flexibility to vary production lots within a production cycle. For that reason most of the current research has been directed to more flexible models.

2.1.3 Time-varying lot size approach

To achieve greater flexibility and overcome the problems faced with a basic period approach, Maxwell (1964) was the first to suggest time-varying lot sizes. Delporte and Thomas (1977) combine a number of earlier research papers to form optimized production sequences from cycle times and product frequencies.

Dobson (1987) uses a similar approach to develop an heuristic that not only solves the feasibility problem encountered in earlier papers, but also spreads machine utilization more uniformly by optimizing idle times between production runs.

In addition to the notations used before, Dobson (1987) introduces variables for production- and idle times within a sequence J.

t production time
u idle time
f production sequence, e.g. $f = (1, 2, 1, 2, 3, 2, 1, 2)$

The actual solution algorithm consists of two stages. First the production sequence f is determined. Then a choice of production- and idle times is made. After a series of substitutions he arrives at the minimization problem P.

$$\text{Minimize } (\sum_{i=1}^{m} \frac{a_i}{x_i})(\sum_{i=1}^{m} s_i x_i + \bar{u}) + (\frac{\sum_{i=1}^{m} b_i x_i}{\sum_{i=1}^{m} s_i x_i + \bar{u}}) \tag{2.1.15}$$

where

$$a_i = \frac{\frac{1}{2} h_i (p_i - r_i)(\frac{r_i}{p_i})}{(1 - \sum_{k=1}^{m} \frac{r_k}{p_k})} \tag{2.1.16}$$

$$b = A_i (1 - \sum_{k=1}^{m} \frac{r_k}{p_k}) \tag{2.1.17}$$

The result of problem P would be a vector x.

By using related work from Roundy (1989), they arrive at the relative frequencies of production n_i. These raw results are used by Zipkin (1991) to arrive at a complete and feasible production cycle. His heuristic uses the set of n_i to sort products into "bins", according to relative frequency. The bins subsequently become the production schedule f.

This schedule might still be infeasible. But as demonstrated by Dobson (1987) before, any schedule can be made feasible by adjusting production times t^j and idle times $\hat{u}^j$.

The heuristics of Dobson (1987) and Zipkin (1991) combined produce near-optimal schedules for the ELSP problem (Moon, Silver, & Choi, 2002).

Up until today, the time-varying lot sizing approach by Dobson (1987) is used as a basis for optimizing the original ELSP. Some successful implementations include a hybrid GA by Moon et al. (2002), tabu search and neighborhood search by S. Raza et al. (2006), as well as Simulated Annealing (SA) (A. Raza & Akgunduz, 2008).

2.2 Sequence-dependent deterministic ELSP

In section 2.1.1, general model assumptions for the ELSP were given. The most unrealistic assumption in this set are Sequence-Independent Setup times (SIS). Researchers have ignored

them for convenience and to easier reach analytically optimal solutions. Probably because it adds considerably to the complexity of any solution method and was proven to be NP-hard, as explained in Pinedo (2008, p. 593).

Nevertheless, in many practical situations they play a vital role and can easily erode the advantage won by using a mathematically more efficient model that ignores them. Several studies presented in Allahverdi et al. (1999, p. 219) found that SDS are especially relevant when a job shop is operated at almost full capacity. Moreover a survey of production managers found that about three quarters agreed on the importance of SDS in *some* production processes, while 15% reported *all* processes requiring SDS.

It was soon discovered that the sequencing of deterministic ELSP is in fact a Traveling Salesman Problem (TSP). That's why literature has closely tracked the progress of TSP and related optimization heuristics. This section will first give an overview of earlier heuristics and limitations made in this area. Then some popular approximate- and heuristic approaches are presented. Moreover the influence and usability of different methods is elaborated.

2.2.1 Earlier approaches

One of the first to examine sequence dependent setup times closer is Maxwell (1964, p. 92). He uses a simple *PxP* matrix $S_{i,j}$ to store setup times. He then goes on to define a range of special $S_{i,j}$ matrices that might occur in practice, like *Equal-To*, *Equal-From*, *Come-Down* or the sequence independent *Equal Entry Matrix*. In his numerical example consisting of four products, he uses a TSP to bring his FC sequence in order. His heuristic then works on top of this sequence and decides on cycle length and time. Though, he doesn't explain *how* he arrived at his traveling salesman solution.

Due to the NP-hardness, researchers in the earlier days focused on special cases of the TSP to arrive at analytical solutions. A popular example is the *one state-variable machine*. A practical example for such a setup would be a size-adjustable iron pressing machine. Before processing a job, the machine has to be put in state A_j. Upon completion of the job, the machine is in state B_j. To start the next job, it needs to be put in state A_{j+1}, which requires an investment of time or money. A solution for such a simplified TSP was first proposed by Gilmore and Gomory (1964) and is also elaborated in Pinedo (2008, p. 84). Unfortunately these kinds of setup matrices are not too common in practice.

2.2.2 Exact approaches

Exact approaches guarantee the best solution available, while a considerable amount of computing power and time. These methods are suitable for smaller problems or to calculate the lower bound and compare it to heuristic solutions.

Dynamic Programming

DP was initially popularized by Bellman and strives to find optimal solutions for small- and medium sized problems. It rests on a simple idea that a problem can be split into a number of subproblems. These are then solved to optimality, which should be easier than solving the primary problem at once. As long as subproblems can be improved, the whole solution can be improved as well.

Bertsekas (2005, p. 18) compares the the principle to a car journey. When the route from Boston to Los Angeles runs through Chicago, finding a better route from Boston to Chicago also results in a better overall route.

A promising approach to solve the Sequence-Dependent ELSP (SD-ELSP) by using DP comes from Buzacott and Dutta (1971). They use a DP heuristic to recursively search for an improved sequence. An initial feasible solution consisting of a total of P products is produced and then improved p products at a time. The size of p is left open for experimentation, but they note that higher values produce better results while adding to execution time. Suffering the curse of dimensionality, the procedure reaches computational limits of that time at around 12 products.

Branch and Bound

Barnes and Vanston (1981) take lease from ongoing research on traveling salesman heuristics and apply a Branch and Bound (B&B) algorithm to the problem. B&B works by subsequently examining *branches* of a decision tree, while calculating *bounds* based on the expected minimum (or maximum) value that can be expected when following a certain branch. Less promising branches are pruned in favor of those offering better bounds. B&B can limit the decision space of a problem significantly, when compared to full enumeration. For the SD-ELSP, the search of a solution for a five-product problem was limited from 82 to 32 states.

One of the drawbacks of B&B is its high computational effort that results in a long running time when solving large problems (R. Tan et al., 1997, p. 630). Compared to random search algorithms the running time was about 20 times as long. In a comparison of different search heuristics, B&B was found to be as much as 10 times larger than GA or SA. It solved smaller problems quite competitively though (K. Tan et al., 2000, p. 319).

2.2.3 Heuristic and metaheuristic approaches

This class of approaches doesn't guarantee the best solution, but usually takes less time to compute. An important distinction can be made between ordinary heuristics and metaheuristics. The former is made specifically to deal with a certain problem, the latter can be used to solve a wide range of optimization problems.

While there exist a range of heuristics to solve the TSP, most SD-ELSP literature uses metaheuristic approaches (Zhu & Wilhelm, 2006).

GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) describes a class of metaheuristic optimizers that can be used to find solutions for problems with large, but finite sample space. Similar to Tabu Search (TS) and GA it also tries to avoid getting trapped in local optima by allowing a degree of non-optimal solutions.

The algorithm works in two phases (Zapfel, 2010, p. 78). In the construction phase candidate elements are assembled in a Restricted Candidate List (RCL), which is evaluated on the basis of a performance function. The top elements from a sorted RCL are then selected at random and added to the partial solution. This concludes the first phase.

In the second phase, candidate solutions are further improved by a local search technique.

GRASP was implemented for a SD-ELSP by Feo et al. (1996). They successfully solve scheduling problems with 20 jobs and get results comparable to TS for bigger problems of 35 to 165 jobs.

Simulated Annealing

Just like a number of other recent optimization concepts, like GA or Ant Colony (AC), SA was inspired by processes occurring in nature. In this case the inspiration comes from thermodynamics and the cooling process of liquids, before they turn into rock or glass. To achieve high-quality results (i.e. the most regular atomic alignment), the cooling process needs to be sufficiently slow.

A global variable for temperature T controls the process. To avoid getting stuck in local optima, at high temperatures, uphill moves are allowed. As temperatures get lower, small improvements are preferred. The process doesn't guarantee the best solution, but only a relatively good solution in a large search space. The complete algorithm is elaborated in Zapfel (2010, p. 112).

R. Tan et al. (1997) tested SA on a 10-job problem to minimize total tardiness and got optimal results in about half the test cases. During the rest of the time their solutions were within 5% of the optimum solution. Kim et al. (2002) compare a neighborhood search to a SA algorithm and find that SA always outperforms the former.

Genetic algorithms

The use of genetic algorithms to solve SD-ELSP is especially promising because it has already been used to successfully solve closely-related TSP (Merz & Freisleben, 1997).

The technique also requires less book-keeping than B&B and Integer Non-Linear Programming (INLP). (Huang & Yao, 2008) By fine-tuning the various settings and crossover algorithms of an GA and combining it with a local heuristics search outcomes can still be improved (Sengoku & Yoshihara, 1998).

This paper also uses a GA to determine the sequence of production decisions. While the detailed workings of a GA are describe in section 4.2, the following part will give an overview of recent work in the field of GA for solving SD-ELSP.

Being a black box optimizer, it doesn't need any knowledge of the underlying problem. This job is handled by the value function that assigns fitness values to the chromosomes it tests. The fitness evaluation might happen by applying a simple formula or by running a simulation program with the input parameters.

Due to this versatility, GA has been used to solve a series of problems related to the ELSP. Huang and Yao (2008) used it to solve a standard ELSP by using the EBP first suggested by Elmaghraby (1978). They use a hybrid genetic algorithm to solve the EBP approach and compare it to two other heuristics. They encode the multipliers k_i in their chromosome by encoding them as a binary string. Next linear ranking normalization is used to store- and rank the chromosomes in a temporary list. A lower objective function value means a higher fitness, because the corresponding chromosome is resulting in lower setup- and holding costs. The reproduction probability of a chromosome increases with higher fitness as compared to the rest of the population.

To avoid losing too many superior chromosomes in each generation, 20% are copied over to the next generation. Next uniform crossover and a random mutation are applied to the chromosome.

The authors then compare their GA to *Power-of-Two* and *Two-Point* heuristics suggested in an earlier paper. The genetic algorithm not only triumphs in solution quality, but also delivers a steady solution quality as utilization increases. This example demonstrates that a GA approach can deliver excellent results, but requires a degree of tinkering on all available parameters and intermediate techniques deployed to reach those results.

Another popular paper (Rubin & Ragatz, 1995) has attempted to minimize total tardiness of jobs under the condition of SDS. The authors compare a rather simple GA to B&B. In comparison to Huang and Yao (2008) they don't use binary, but integer encoding in their chromosomes to avoid infeasible solutions. To generate new candidate solutions, a crossover- and random mutation algorithm are deployed.

They find that their GA outperforms B&B at problems with more than 25 products. In this case they truncated the B&B and used the best solution found so far. When the B&B was allowed to run to completion, the GA matched its performance closely.

Experimentation with parameters and the additional use of heuristics to improve candidate

solutions seems to pay off when using GA. This was also discovered by Miller et al. (1999). The authors attempt to minimize setup-, inventory- and backlog cost of a press shop for plastic parts, which consists of a single facility facing SDS. Demands are deterministic and arrive through a Material Requirements Planning (MRP). The daily production schedules have to be generated for around 15 products.

Candidate solutions are encoded in integer arrays, storing the position of each item in the production sequence. Mutation and crossover are organized in a similar fashion as in the previous papers. Additionally, Miller et al. (1999) incorporate a hill-climbing technique based on the Wagner-Whitin algorithm to improve every new generation. This improves feasibility and reduces setup cost.

In a numerical study they observe an improvement of about 35% over a standard GA. The hybrid approach seems to be especially effective in initially reducing setup costs. These results confirm the previous observation that GA are well suited for the optimization of SDS scheduling problems. But at the same time they require more adjustments than other techniques. Mixing them with heuristics also seems to be beneficial, but requires more initial work.

2.3 Stochastic ELSP

Deterministic ELSP and their sequence-dependent extensions are useful when scheduling a pre-defined set of jobs in a controlled environment, like daily work assignments on a made-to-order production facility. But in many situations, demand won't be known exactly in advance. This will, for example, apply to businesses that sell directly to consumers and face an uncertain demand.

This section will relax the final assumption made in 2.1.1, i.e. known and constant demand rates. Stochastic demand rates are a big challenge for production- and inventory planning, because inventory- and production quantities need to be adapted to an uncertain demand rate. Moreover no precise outcome of a certain policy is available, but results will always be within an interval. Winands et al. (2011, p. 2) note that findings from ELSP are still useful as a basis for planning in a stochastic environment, but also emphasizes some major limitations.

First, a rigid cyclic production plan, as is usually suggested for any cyclic ELSP is not applicable any more due to changing demand rates. Second, inventory plays a more important role to guard against spikes in demand. Moreover, it is not economical to cover 100% of the demand due to these fluctuations in demand, rather the goal will be to cover demand as far as economical, given certain holding costs.

As stated in an earlier research paper by Sox et al. (1999), SELSP control policies consist of two critical elements: lot sizing and sequencing. The former is concerned with adapting

production- and inventory levels to changing demand situations. The latter is about deciding on the actual production order. (this will become even more relevant in section 2.4.)

2.3.1 Production sequencing in SELSP

Sequencing policies of SELSP can be classified in two broad categories (Sox et al., 1999). Cyclic schedules keep a fixed sequence and only vary lot sizes. Their dynamic counterparts reevaluate sequence *and* lot sizes after each decision period. Cyclic schedules were suggested in the following works:

Cyclic product sequencing

Gallego (1990) combines earlier research from Zipkin (1991) to construct a full control policy for Brownian Motion distributed demands. He initially uses expected means of the distribution to construct a production sequence and cycle length T, as shown in Zipkin (1991). He then moves on to calculate produce-up-to levels for a given demand variability. Last he uses Monte Carlo simulation to get estimates for required safety levels. When adding random demands to his initially deterministic model, he finds that average costs are 23% higher. This is mostly a result of safety stocks.

An early model, which is related to the technique later described in this paper comes from Bourland and Yano (1994). They us a two-level hierarchical solution approach to the problem. At the higher *planning* level, parameters like idle time and safety stock are set. At a lower *control* level these parameters are then used to guide operational decisions.

In terms of sequencing, they evaluate a pure rotation sequence (sometimes also referred to as common cycle), as well as a fixed cycle sequence, which they determine by using the heuristic for ELSP, as presented in Dobson (1987). Their model consists of 4 products. The simulation environment GAMS is used to test control policies. While they initially fix an idle time of 10% to account for unexpected demand events, they find that optimizing idle time as yet another decision variable decreases cost significantly. When using a fixed sequence, calculated by Dobson (1987), higher setup costs are more than offset by decreased holding costs which results in a 4% decrease in cost.

Dynamic product sequencing

For stochastic environments, dynamic policies should be preferable because they adapt to changing demand quantities. An early approach that already relies on simulation modeling for benchmarking comes from Vergin and Lee (1978). They realize the importance of realistic demand modeling. In their paper they use a compound (stuttering) poisson distribution.

Time between demand events is modeled as poisson distribution. Units of demand come from a geometric distribution.

They continue to compare a number of control policies. Rule number one and two are purely deterministic rules they are using on their stochastic problem. As expected they don't deliver the most satisfactory results. Rules three to six are based on Magee. He suggests to only control and adjust inventory levels, instead of actual production. This idea–though poorly executed–can be viewed as a predecessor to more fine-grained s,S policies. Magee's original formula to calculate S_i is

$$S_i = \frac{2d_i(1 - \frac{d_i}{p_i})}{\sum_{i=i}^{m} d_i(1 - \frac{d_i}{p_i})} \tag{2.3.1}$$

Production is continued until one of these conditions applies:

1. inventory of a product runs out.
2. inventory of product i currently being produced, reaches S_i

Vergin and Lee (1978) expand this rule to produce up to a minimum number of days, to avoid short production runs and to restrain inventory buildup caused by the original set of rules. In their tests, the original rule has problems with excess inventory when demand is less than production capacity.

The authors then use a simulation to benchmark the six decision rules. Rules number one and two perform quite poorly in a stochastic environment. Magee's original procedure has problems with inventory, as explained above. The extensions to Magee made by the authors performs better.

Though the approach made by Vergin and Lee (1978) is rather simple, they demonstrate that deterministic production rules can't just be applied to stochastic situations. They also find that a controlling production through inventory levels is a more effective approach than setting production rules.

Another approach, which is slightly related to Löhndorf and Minner (2011), can be found in Graves (1980). The author assumes a stationary distribution and periodic review. Setup cost are incurred, when the production facility switches from one product to another or wakes up from an idle period. The objective is to minimized expected cost per period, consisting of setup, inventory holding and backorder cost.

He then develops an heuristic based on the expected value of a Markov decision problem and notes that an optimal policy must be based on two numbers, denoted as I^* and I^{**}. A point when production starts and another one at which it finishes. To find suitable values for the

two parameters, the state space is limited to a number of discrete intervals and then examined by policy iteration.

Subsequently the author expands his suggested cost function to cases with multiple products. As long as one product decision doesn't conflict with another, products can be scheduled equivalent to single product cases. When more than one product is scheduled to be produced at the same time, contradictions occur. In such cases a resolution mechanism is needed. Graves (1980) initially suggests to use the value function of the one-product case as a benchmark.

This method can resolve product conflicts, but does not anticipate them. To account for this complication, the author introduces composite products. These are similar to aggregate products in linear programming an help to avoid conflicts by pushing the control policy to use additional inventory.

A numerical study of five different versions of the control policy reveals a good performance for the composite product-approach. Heuristics using lowest runout time perform not as good. The composite product-heuristic is only improved by a preemption policy that delays production in case another product reaches a critical level.

Leachman and Gascon (1988) present another interesting approach inspired by a manufacturer of fashion items. The authors combine a number of techniques from ELSP scheduling to dynamically plan upcoming production cycles.

Their method is based on two common cycles–a target cycle and operational cycle. Initially the target cycle is calculated from moving averages of historical demand patterns. Then they use the common cycle heuristic proposed by Doll and Whybark (1973) to find appropriate T^* and multiples k_i.

The next step is to find the length of an operational cycle based on Runout Time (RO) RO_i. This is a measure of when production needs to start to avoid a stockout situation. Figure 2.1 shows a (not so typical) production system with RO. Demand is assumed to be linear over the planning period. The black bars denote production time. The white bars represent depletion of stock. Items 1 and 2 have positive slack available. Production for item 3 should therefor start earlier because there is negative slack between item 3 and 4. If the total (positive and negative) slack in the production cycle is negative, the cycle is not feasible.

Notations introduced in section 2.1.1 are still being used. In addition, these values are needed as well:

RO_i	runout time of item
I_i	inventory level of item at start of period
t_i	time period index
ss_i	safety stock
ch_i	changeover (setup) time

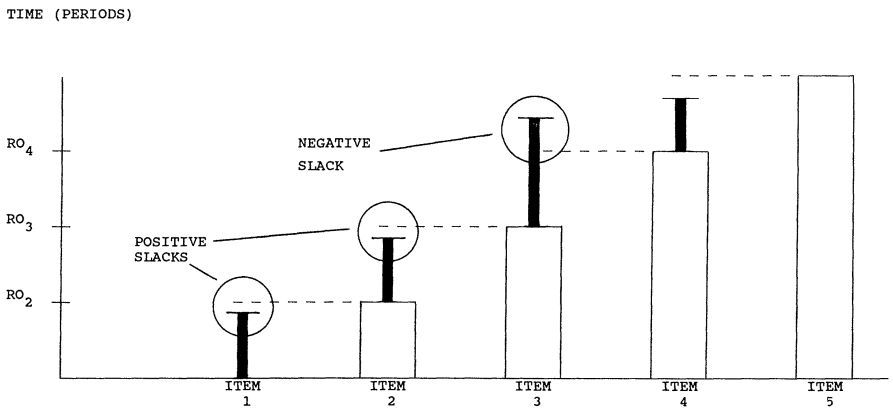


Figure 2.1: Production system out of balance
(Leachman & Gascon, 1988, p. 380)

Without taking stochasticity into account, *RO* of item i can now be calculated.

$$RO_i = t' + \frac{1}{d_{i,t'+1}}[I_i - ss_i - \sum_{t=0}^{t'} d_{it}] - chi \tag{2.3.2}$$

ROs are used to get a production sequence, by sorting them from lowest to highest.

$$RO_2 \leq RO_3 \leq \cdots \leq RO_n \tag{2.3.3}$$

By using ROs as bounds and (2.1.14) from Doll and Whybark (1973), the cost effective T can be found.

$$T = \max\{\frac{mP_1}{k_1 D_1}, \min\{T^*, \min\frac{RO - j - \sum_{i=1}^{j-1} c_i}{\sum_{i=1}^{j-1} \frac{k_i d_i}{P_i}}\}\} \tag{2.3.4}$$

With cycle lengths T fixed, produce-up-to-levels can be found.

$$S_i = ss_i + \frac{T_i d_i (p_i - d_i)}{p_i} \tag{2.3.5}$$

In the next step Leachman and Gascon (1988) use the standard deviation of forecast errors to construct a confidence interval of *RO*.

$$RO_i \pm Z\frac{\sigma_{i1}}{d_{i1}}(RO_i - 1)^{\frac{1}{2}} \tag{2.3.6}$$

The strengths of the methodology presented above are relative simplicity and the application of concepts familiar to many practitioner. The drawbacks are poor performance with more sporadic demand patterns, since the adaption of *RO* always assumes a normal distribution. This would–for example–be unsuitable for compound distributed demands.

In Leachman et al. (1991), an improvement to the original heuristic that improves the calculation of production time and yields a cost reduction of about 1-7% is published.

The work of Sox and Muckstadt (1997) also deals with dynamic scheduling in stochastic demand situations with deterministic holding-, production- and setup cost. They find that their model is most suited for situations with sporadic demand. The Linear Programming (LP)-model they suggest relies on historical forecast data to plan the current period. Demand forecasts are updated after each period and then modeled as distributions, rather than point-forecasts.

They solve their model by using Lagrangian relaxation to decompose it into different subproblems and then solve it by using B&B. While admitting that their procedure doesn't eliminate the complexity of the overall problem, it got reduced to a manageable size.

The model gets tested with data from an aerospace manufacturer. Demand is modeled as negative binomial distribution (due to high demand variation) for 5, 10 or 15 products. They find that their solutions are about 5% above the lower bound. This number increases at higher product counts and high utilization. They also find that setup times make the problem more difficult to optimize, because their binary variables for SDS have a big influence on the outcome.

In conclusion, the approach presented by Sox and Muckstadt (1997) works well with smaller problems at relatively low utilization that don't have SDS. The possibility of using a custom demand distribution is also a benefit.

As mentioned before, the the SELSP can be solved by using a variety of techniques. New optimization techniques are usually applied to the problem sooner or later. Recently a number of researchers was inspired by Artificial Intelligence (AI) methods, more specifically Reinforcement Learning (RL).

Paternina-Arboleda and Das (2005) use multi-agent RL to dynamically find production decisions. Their work is inspired by a paper that suggests a similar approach to improve routing of Internet Protocol (IP) packets.

The authors solve the problem by first deriving a mathematical model of the problem. Demand arrives in random intervals and inventory is constantly monitored. Whenever a base stock level of R_i is reached, their agents make a production decision.

Next a RL algorithm to either lookup a known action a that minimizes the expected cost of the decision $R_m(i,a)$ or choose another action from the set A_i/a. After the decision, simulation of the system is continued and the immediate cost $c_{imm}(i,a,j)$ for the state-action combination (i,a) is recorded.

Storing all the state-action combinations for a bigger production problem is not realistic. The authors avoid this problem by introducing a Artificial Neural Network (ANN) scheme for approximation the value function for unknown states-action pairs. They are using a Least Mean Squares (LMS) algorithm, which assumes linear relations. After each decision made by the RL algorithm, the corresponding action LMS neurons are updated.

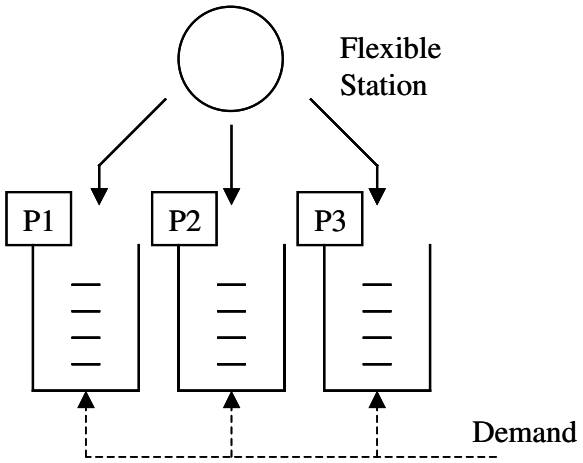


Figure 2.2: *Production system with inventory*
(Paternina-Arboleda & Das, 2005, p. 396)

The production process Paternina-Arboleda and Das (2005) use is illustrated in figure 2.2 and quite similar to the setup used in this paper. A flexible station is feeding into a Finished Goods Inventory (FGI). Demand is satisfied from inventory and follows a lognormal-distribution. The goal is to keep inventory and setup costs low, while at the same time satisfying all customer demand.

The authors use numbers from an earlier paper to validate their simulation model. They continue by keeping the base stock levels of the reference paper and compare their RL schedule to an existing ABAC[1] schedule. Their RL schedule reaches about the same average cost as a fixed cycle policy. The improvements are not conclusive when taking into account deviations of ±3% between the simulation models. (Paternina-Arboleda & Das, 2005, p. 398)

Subsequently the authors attempt to use a fixed RL policy to search for better base-stock levels. They limit the search space by only considering slots of ±5 units. Again, the improvements of the original policy are inconclusive for a case with low variability and only marginal for a sample with extremely low variability, after accounting for possible differences between the simulation models in use.

[1] ABAC describes a fixed cycle that starts with product A and finishes with product C.

Löhndorf and Minner (2011) deal with a similar SELSP and compare the performance of an Approximate Dynamic Programming (ADP) approach to a fixed cycle- and base stock solution. Similar to previous works, their model consists of a single machine capable of producing a number of different products. Setup- and production times are deterministic. Finished goods are put in inventory, from which demand is satisfied. Demand follows a stuttering (compound) poisson distribution.

The production scheduling process is modeled as Semi-Markov Decision Process (SMDP). Decisions are made after demand events or after production finishes. States are defined by the setup state of the machine and inventory states.

They find that ADP works well with as little as three products, but is outperformed by preemptive FC rules based on global policy search at 5 to 10 products.

2.4 Sequence-dependent stochastic ELSP

Up to now, a variety of approaches of solving SDS or SELSP that exist in literature have been reviewed. This chapter relaxes the two final assumptions from 2.1.1 at the same time. Demand arrives randomly according to a probability distribution and setup times are dependent on the last product that was produced.

Research concerning such a case is very sparse compared to the multitude of material available for SELSP or SDS cases. Ashayeri et al. (2006) present a case study involving minor and major setup times that are sequence-dependent between product families. To determine the production sequence, a TSP or *experience* is suggested. Frequencies and order-up-to levels are determined by using a simple fixed cycle heuristic with a cycle length T^* and multipliers k. The paper doesn't present any calculations, but point to another case study by Strijbosch et al. (2002).

The authors of this paper deal with similar issues as Ashayeri et al. (2006). A TSP is suggested to order sequences. Production sequences are found by using an iterative approach starting with EOQ, to calculate base cycles and multipliers. This technique is very similar to Doll and Whybark (1973).

Both papers are more concerned with implementation and strategic improvements to logistics and inventory and don't suggest any new techniques for solving the problem.

3 Problem description

This chapter gives an initial overview of situations that are faced with SDS under stochastic demand and could benefit from improved production planning. Because research in this direction is sparse, it should be assumed that many more situations exist.

The second part of the chapter dissects the problem at a high level and elaborates the individual sub-problems that need to be solved.

3.1 Practical relevance and applicability

3.1.1 Packaging in the pharmaceutical industry

A case study introduced by Strijbosch et al. (2002) deals with a Dutch pharmaceutical company. The firm is evaluating a system of centralized inventory for three Northern European countries. Currently inventory is held independently in all three countries by local distributors. Orders flow from the end user to the distributor to the manufacturer. While there is sufficient scale with the production of the actual product, packaging remains a challenge due to different languages and package shapes in the target countries. The distribution system currently in use is displayed in figure 3.1

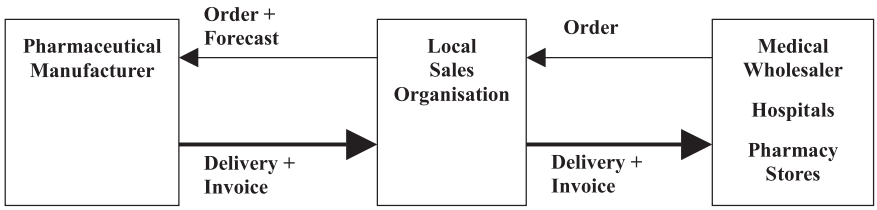


Figure 3.1: Current supply chain of a Dutch pharmaceutical company (Strijbosch et al., 2002, p. 551)

Orders are collected one to three weeks in advance and then scheduled for efficient production. This–in fact–transforms stochastic demand in a deterministic ELSP. An efficient production schedule can be created after all the orders have arrived. While such a system makes planning easier, it brings some disadvantages for the supply chain. Orders arriving shortly after a production plan has been made, need to wait for the next period. This artificial bottleneck causes high lead times and requires increased inventory with the distributors.

Packaging of all articles happens on the same machine. Articles are organized in families. Switching within families is called a minor setup. These small adjustments take about ten minutes. Changing from one family to another is called a major setup. This includes extensive cleaning and reconfiguration of the machine. Such a procedure takes at least 75 minutes or more.

The authors of the case study propose a fixed cycle time approach to arrive at a cyclical production schedule. Additionally, such a move would be accompanied by modifications in the supply chain. Instead of keeping stocks with each distribution partner, shipments would go directly from the manufacturer to the final point of use (e.g. hospitals or pharmacies) via a third party distributor, like UPS or DHL. An outline of the revamped supply chain is presented in figure 3.2.

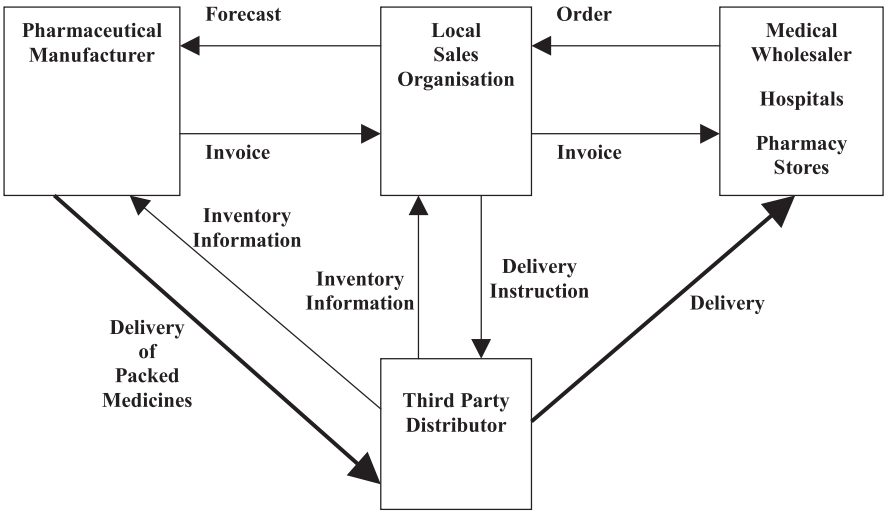


Figure 3.2: Proposed supply chain of a Dutch pharmaceutical company (Strijbosch et al., 2002, p. 552)

Such a system would combine three stock points into one. This would give the producer more control over product shipments and reduce total inventory through the portfolio effect. In addition, the local sales partners can focus on their core activity of selling, instead of maintaining a parallel logistics infrastructure in each country. (Strijbosch et al., 2002, p. 548)

In such a setup, it would be a great advantage to switch from ad-hoc cycles based on demand to optimized cyclic schedules. In the current example, products can be grouped into families (A, B and C). Each family consists of articles. Family A might have 5 articles from a_1 to a_5.

The combination of techniques proposed in this work could be used to arrive at an optimized production- and inventory policy.

3.1.2 Production planning in the pre-Deco industry

Another case study by Ashayeri et al. (2006) deals with a similar problem. The subject under study is the production process of a pre-Deco manufacturer. Pre-Deco products are sold in do-it-yourself (DIY) markets along with other crafting material. Willing consumers use them to skillfully craft presents or decorate things, like greetings- and birthday cards. The company currently produces about 250 different products that are shipped in 1000 different Stock-Keeping Units (SKUs).

The product range can be categorized by sales volume in high- and low volume products, or by production process in powders, ready-mixed, liquids and others. The first three products are manufactured internally. The last category includes some slow-moving products or products that require specialized technical expertise. This requires them to be manufactured externally.

Similar to the previous case study, SKUs are shipped to independent distributors, who organize sales and logistics in national European markets. These autonomous partners each keep separate safety stock- and demand forecasts. They also determine production time and batch size by using a simple time-supply rule in connection to the next possible production date at the factory. Whenever inventory is below safety stock at the next possible production date, a production cycle is initiated. So reorder point R equals safety stock s_i plus forecast demand during waiting- and production time p. Production plans are drafted at the end of each week. Raw material and workers are also hired accordingly.

This production system clearly has a number of drawbacks. Production costs are usually higher than expected and inventory levels at the independent distributors (for which the factory is responsible) are too high. The authors of the case study identify the following areas for improvement: First the production plan should be more stable to make procurement of intermediate material easier. Next, by developing an easy-to-understand inventory model, the number of changeovers (i.e. setup costs) needs to be reduced.

Similar to the previous example, reorganization and an enhanced cyclical production cycle could bring big benefits to this supply chain. Figure 3.3 illustrates the proposed planning cycle. Initially base material is procured, then products are produced in a fixed cycle, starting with product D, then A, C, D, C. Production quantities are based on probability distributions derived from locally collected sales data.

The distinction between high- and low volume products is a clear indicator that a fixed production cycle might be appropriate. The size of the problem requires a technique that can scale to a large number of products without suffering from the curse of dimensionality.

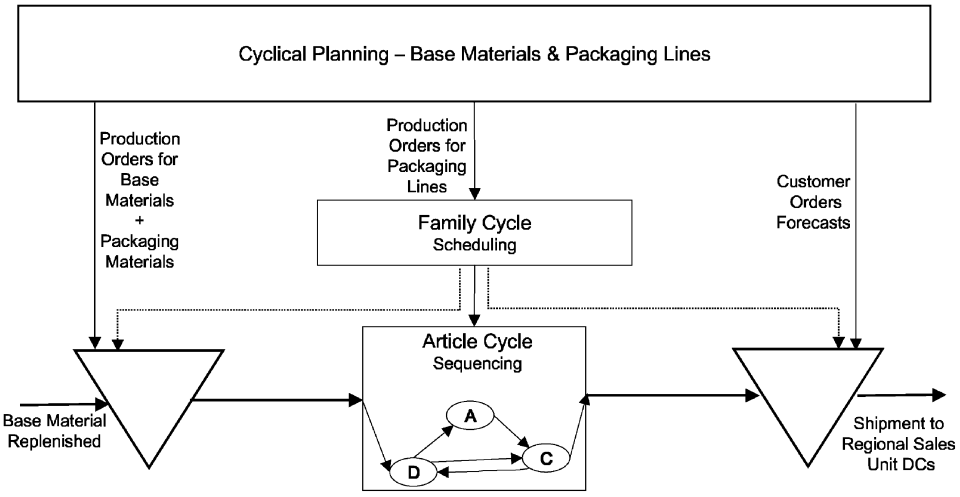


Figure 3.3: Proposed cycle planning for pre-Deco production (Ashayeri et al., 2006, p. 718)

3.2 Critical elements of the production plan

Arriving at a usable production plan to attend to the problems described in the previous chapter requires the successful unraveling of a series of sub-problems. In short these are production sequences and lot sizes. The following classification is adopted from Winands et al. (2011, p. 3), who use it to classify SELSP literature.

3.2.1 Production sequence

The production sequence describes the order in which different products are manufactured. A *sequence* consists of an *order* and a *frequency*. E.g. the sequence $Q = \{A, C, A, D, B\}$ has four products from A to D. The frequencies would be $f = \{2, 1, 1, 1\}$. Frequencies simply specify the number of times a product appears in a single production run. As was mentioned in the second case study, high volume products will show up more often in a cycle than low volume products. On the condition that setup costs are not too high, it will usually be cheaper to produce high-volume products more than once per cycle to reduce inventory.

A useful initial classification of production sequences are their *order*. No matter how often a product appears, the sequence is either *fixed* or *dynamic*.

In the class of fixed sequences, one distinction can be made on the basis of product frequencies. When every product within the sequence is only produced once, the cycle is called a *pure rotation cycle* or *common cycle* because all products share the same cycle and cycle length.

Sequences that contain one product more than once are referred to as *basic period approach*, as explained in (Elmaghraby, 1978). While all products still share a basic period as common denominator, their individual cycles are determined by the value of a multiplier k_i. All multipliers k_i together form the frequencies f.

A further useful distinction can be made on the basis of total cycle length. (Winands et al., 2011, p. 3) A *fixed cycle length* means that every cycle has the same length, but not necessarily the same production quantities. This can be useful, when external distributors rely on a regular production cycle. In such a case any modifications need to fit within the cycle.

As opposed to a fixed sequence with a fixed cycle length, there can also be a fixed sequence with dynamic cycle length. These are especially useful, when demand is unknown, but the production order should be kept the same because of dependences, like SDS.

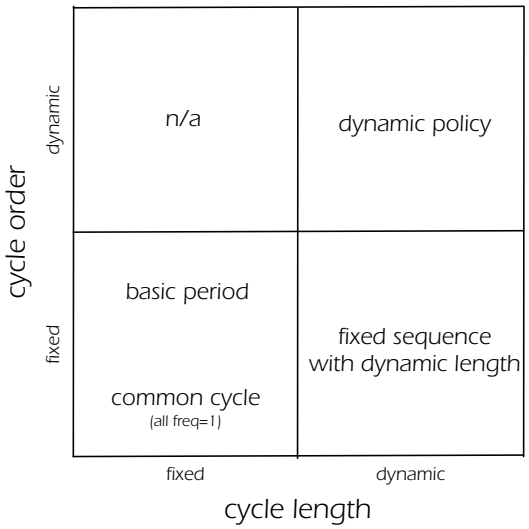


Figure 3.4: *Classification of production sequences*

Dynamic sequences are not as well researched as fixed sequences, but could theoretically create the best production schedules in a stochastic environment. Production order is always determined on an ad-hoc basis, derived from current inventory- and setup states of the facility. This approach was tried by Löhndorf and Minner (2011) in the form of ADP. The implementation performed well for very few products, but performed poorly with more than three products.

A summary of possible sequencing policies, according to cycle length and order is given in figure 3.4.

3.2.2 Lot sizes

After the production sequence has been determined, the next step is to fix production- and inventory quantities. In a deterministic system lot sizes are easy to determine, once cycle times are known, because demand for a certain time span is known in advance.

In a stochastic environment, the task is not so easy. The two most common frameworks to do so are the (R, Q) and (s, S) policy. The benefit of using a framework is that the only problem left to solve is on deciding on the parameters. Chapter 2 has already introduced a wide range of

solution approaches that range from using heuristics operating on mean demand to dynamic approaches working on current inventory levels.

These solutions might work well on SIS. SDS add another problem dimension though. No matter which control policy is used, as soon as products are skipped or preempted, the most efficient cycle/sequence is left and inefficiencies arise.

An optimal solution of the SD-SELSP would mostly stick to the most efficient sequence. In case of low inventory situations, it would only leave the cycle, if by doing so, it can avoid lost sales cost in excess of the additional setup- and holding cost caused.

4 Solution methods

Chapter 3 introduced some practical examples for business situation, this work is trying to improve, as well as the sub-problems that need to be solved first. Namely, finding a desirable sequence, the corresponding production frequencies, an inventory- and setup cost-minimizing order- and base stock levels.

To solve each of these three problems, there are a number of different tools, techniques and approaches available. This chapter will first introduce the necessary building blocks to solve individual problems and later assemble them candidate policy combinations.

The chapter starts with introducing the global optimizer from Hansen and Ostermeier (2001), which is used for setting base stock levels and some parameters of other optimizers. Then a multi-criterion GA is proposed to solve the TSP and produce a balanced sequence. For benchmarking the GA sequence is compared to a greedy next-neighbor policy.

4.1 CMA-ES

The Covariance Matrix Adaption Evolutionary Strategy (CMA-ES) is a stochastic search algorithm that can minimize a non-linear objective function by sampling it and internally constructing a multi-variant normal distribution to derive new candidate solutions. (Hansen & Ostermeier, 2001)

It doesn't require any knowledge about the search domain, but uses the function values of evaluated search points. A minimal function value should be found in as little evaluations as possible.

4.1.1 Multi-variate normal distribution

The core of the CMA-ES is a multi-variate normal distribution $N(m, C)$. m denotes the distribution's mean value. C stands for a covariance matrix of size n x n. The matrix contains the covariances between elements. In CMA-ES the multi-variate normal distribution is used to generate new search points, evaluating their values and updating the distribution generation after generation. (Hansen, 2006, p. 78)

The objective of the search algorithm is to fit the search distribution to the contour lines of the objective function. This adaption process consists of two steps. First a new mean of the

search distribution is selected by taking a weighted average of μ selected points from the current sample. Points are selected on the basis of their rank after evaluating the objective function. (Hansen, 2006, p. 79)

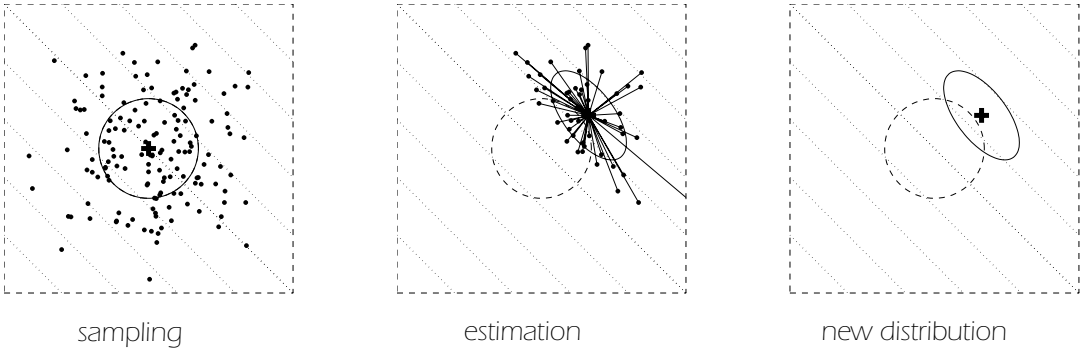


Figure 4.1: Adaption- and sampling process in CMA-ES
(Hansen, 2006, p. 82)

Figure 4.1 illustrates a single adaption step of the multi-variate normal distribution. Initially the search space is sampled and a new mean and covariance matrix is calculated. This shifts the distribution and should produce a better sample of candidate solutions in the next generation.

4.1.2 Adapting the Covariance Matrix

In the next step, the covariance matrix C is updated. Initially C is derived from a single population of the first generation. When the population is small, this estimate is likely to be unreliable. To improve this, an adaption procedure that works with multiple generations is used.

Similar to estimating the mean of the normal distribution, a weighted estimator over several generations is used to construct the next covariance matrix. To account for an improved position of the search distribution over time, more recent generations are assigned a higher weight, through the application of exponential smoothing. A parameter c_{cov} is used to direct the weight of newer generations. Similar to the mutation rate in GA this value is necessary for a balance between fast learning and a stable covariance matrix. (Hansen, 2006, pp 82-84)

4.1.3 Step size control

In addition to a weighted adaption of the covariance matrix, CMA-ES also employs cumulative step size adaption to arrive at an optimal learning rate. For this the length of the evolution path is evaluated on the basis of its length. If the evolution path is long, the steps that have been taken towards the optimum are small and the distance could also be covered by longer steps in the same direction. As a result the step size is increased.

When the evolution path is short, small steps in different directions cancel each other out and the step size should be decreased.

As a measurement, whether an evolution path is long or short, the expected step size under random selection is used. (Hansen, 2006, pp 87)

4.1.4 Algorithm

To illustrate the concepts presented in this section, a simplified version of the CMA-ES is displayed in figure 4.2.

```
#preparations
set dimension, fitness function, stop criteria, start values, ....
initialize dynamic strategy parameters and constants

#generation loop
while (stop criteria != true) {
        generate and evaluate offspring
        sort results by fitness and compute weighted mean
        adapt covariance matrix
        update dynamic strategy parametrs
        adjust step size
}

#finish
return best point of last generation.
```

Figure 4.2: Pseudocode of CMA-ES
Compare original Matlab code in Hansen and Ostermeier (2001, p. 33)

The CMA-ES is a robust optimizer that has the ability to adapt to a large number of situations. Therefor it should work well for the optimization of base-stock levels, which are based on an unknown and noisy value function.

4.2 Genetic algorithms

GAs belong to the class of directed random search heuristics. They were inspired by the idea of natural selection and have moved many of the terms used in biology to the field of AI.

This section first gives an overview of the biological background that served as inspiration for GAs. Then the encoding of candidate solutions in chromosomes and basic steps in a GA are discussed. The last part deals with multi-criterion GAs.

4.2.1 Biological background

Genetic optimization, as well as other evolutionary algorithms are based on a model of natural, biological evolution, going back to Charles Darwin's theory of evolution. It explains the adaptive change of a species by natural selection. In addition to this constant evaluation of a species' features by its environment, new variations are introduced into the Deoxyribonucleic acid (DNA) by random errors occurring during the copying process. If these mutations prove beneficial in the current environment, they retain and spread throughout the population.

The main driver of selection is the production of offspring. When both parents have successfully survived in their respective environment, their genetical traits life on in their offspring. The concept of "fitness" is only evaluated indirectly. In simulated genetic optimization, fitness is measured directly through a value function.

An important concept in genetics is the distinction between *genotype* and *phenotype*. The former describes an individual's genetic information, as is encoded in its DNA. It is passed on from its parents and remains fixed. The latter means an individual's physical features that can be described as realization of its genotype. An individual might have a potential for blue eyes and a potential for green eyes in its DNA. It can not be known in advance, which one will materialize.

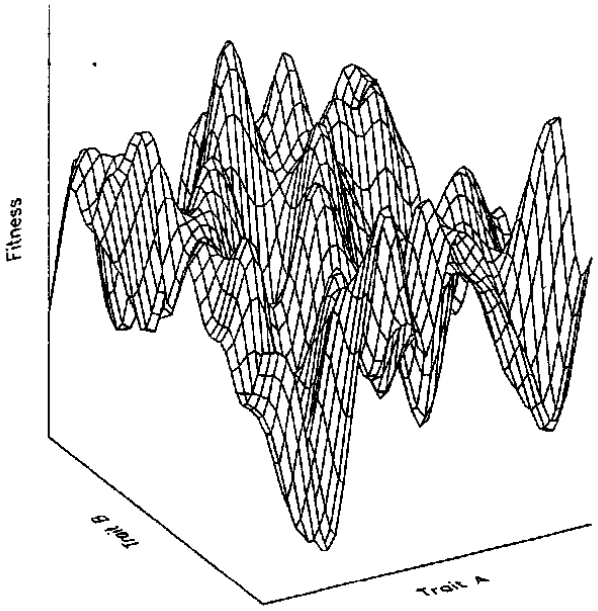


Figure 4.3: Example of a fitness landscape with two traits (Bäck, 1996, p. 10)

The collective genetic information of a population is described as gene pool. It consists of all the successful (surviving) individual's genotype. Over time, as only individuals well adapted to their environment survive, the gene pool adapts to the environment. This is sometimes described as adaption. A popular metaphor for this is Wright's model of a fitness landscape

(see figure 4.3). Different trait combinations result in different levels of fitness in a rocky fitness landscape. Note that this landscape changes over time and formerly useful traits become obsolete.

To prevent early stagnation on a local optimum, genetics has some built-in functions that allow a population to cross a fitness valley, namely random mutation and genetic drift. (Bäck, 1996, ch. 1.1)

4.2.2 Representation through chromosomes

As explained in Bäck (1996), in living organisms genetic information is encoded in DNA. The details of this process are not relevant for optimization applications, but their logical structure is. Starting from the top, an individual's genetic information is encoded in its genotype. A genotype is mostly an abstract word for genome, which contains the same amount of information. Below a genome, there are a number of chromosomes (23 for humans). Further below, two units follow that are mainly important for transcription and reproduction and are generally not used in GA. The important point is that a chromosome consists of a number of genes, several thousand in nature, a few dozen in a typical optimization application. Genes use four nucleotide bases to encode information. A gene contains enough information to encode a single protein. So changing one gene could be interpreted as changing a single trait in an individual's genotype. For a complete hierarchy see Bäck (1996, table 1.2)

For optimization applications the most important units of measurement are the gene pool, a chromosome and a gene. The first describes all the candidate solutions available at a single moment. The second means one complete solution to a problem and the last describes a single parameter of a solution.

4.2.3 Main steps of a genetic algorithm

First an initial population needs to be created. This can happen through random sampling or by using a heuristic. For the best performance, the population should cover the search space uniformly. If there is a bias in the population towards a certain area in the search space, the results might not be optimal.

After the population has been initialized, the first round of evaluation is started. Every candidate solution is evaluated on the basis of a pre-defined fitness function. This could be simple function or a simulation model. (Zapfel, 2010, p. 125)

Once candidate solutions have been evaluated, a procedure for selection is required. In nature selection happens through survival of the fittest. In an optimization environment, a simple ranking is sufficient. Only the fittest X% of each generation are retained and carried over into the next generation.

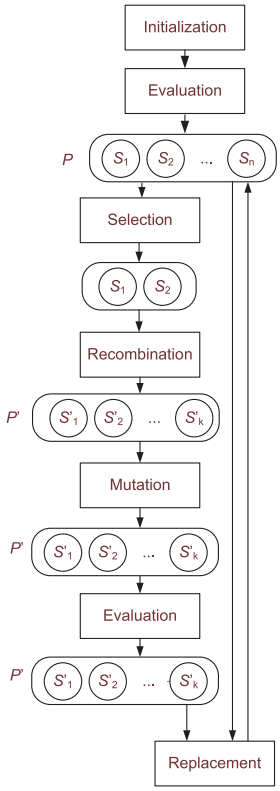


Figure 4.4: Solution process of a genetic algorithm (Zapfel, 2010, p. 124)

The next step in a GA is recombination. Successful solutions are combined to create new ones that exhibit favorable traits of both parents. To keep a higher degree of variety in the gene pool and avoid getting trapped on local optima, random mutation is applied to some of the new individuals as well. (Zapfel, 2010, p. 125)

After the next generation has been created through recombination and mutation, it is evaluated. Then the fittest part of the population is used as basis for the next generation. This process continues, after a stopping criteria is reached. This might be a certain number of generations, or a pre-defined fitness value. Figure 4.4 summarizes the main steps required for a GA.

4.2.4 Multi-criterion evolutionary algorithm

An additional problem encountered in this work is mult-criterion optimization with GA. This problem can be solved in a variety of ways. Ghosh and Dehuri (2004) provide a good overview of available methods. They note that optimization problems with more than one objective function don't have a single optimal solution, but a range of solutions of equal quality. This is called Pareto optimal front. The solution can't be improved in one regard without making it worse to another regard. Figure 4.5 presents a Pareto optimal front with three solutions p, q and r on it. All of these solutions are optimal, despite their objective function being suboptimal when observed in isolation. The main difference with multi-criterion problems is the combination

of a number of objective functions.

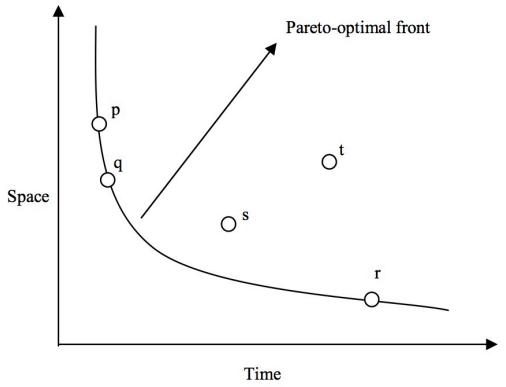


Figure 4.5: Pareto optimal solutions
(Ghosh & Dehuri, 2004, p. 40)

The authors present a number of solution approaches. The one used in this paper is one of the simplest and easiest to use, calls Weight-based Genetic Algorithm (WBGA) Each objective function f_i is multiplied by a weight w_i. The sum of the weighted functions is then used to arrive at the chromosome's fitness.

4.3 Closest neighbor heuristic

The Nearest/Closest Neighbor (NN) is a TSP-specific heuristic that can easily adapted to solve SD-ELSP instances. It mimics human behavior, in that it starts at a random point and always chooses the nearest city. Earlier research has shown that it keeps the tour within 25% of the lower bound. (Matai, Shing, & Mittal, 2011) This didn't apply to sequence-dependent setup costs as was observed in this paper.

```
1 select random city
2 while (unvisited cities left)
3         find nearest city and go there
4 return to first city
```

Figure 4.6: Pseudocode of closest neighbor heuristic
Compare (Matai et al., 2011, p. 13)

4.4 Base stock inventory policy

This section introduces the two most commonly used frameworks for inventory control. Both policies work under stochastic, as well as deterministic demand.

4.4.1 (R, Q) policy

When using this policy, inventory is reviewed on a periodic basis, every T periods. Whenever the stock level drops below a fixed reorder quantity R, a fixed quantity Q is ordered. As this policy is closely related to batch processing, it is sometimes denoted as (R, nQ). This means that multiple batches are ordered. (Axsater, 2006, p. 48)

The main weakness of this policy is its lack of flexibility in adapting the value of Q on the basis of available inventory. If demand is exceptionally high, an order of Q might not be enough.

4.4.2 (s, S) policy

This policy is similar to the previous one. The main difference is that in case of inventory falling below reorder point s, it is replenished up to maximum level S.

For systems that operate on a flexible production quantity, using the (s, S) policy offers a slight advantage. In practice the difference is small and an (R, Q) policy with a fixed batch size is sometimes easier to use. (Axsater, 2006, p. 49)

5 Model and control policies

This chapter first explains the assumptions, this model is based on and why they were used. Later the implementation and the inner workings of the model's transition function are explained.

5.1 Assumptions

In section 2.1.1, the assumptions found in the first ELSP models were presented. They were quite restrictive and unrelated to many production planning problems found in practice (see chapter 3).

Building on the assumptions introduced before, this model presupposes the following:

1. The machine can only produce one product at a time.
2. Inventory holding cost is directly proportional to the amount of inventory.
3. Production costs- and time are proportional to the quantity produced.
4. The sum of production- and setup times does not exceed the total capacity of the production facility.
5. Setup cost and times are *dependent* on the production sequence.
6. Each product has stochastic demand rates that follow a known (compound) distribution.

5.1.1 Production

The first assumption is quite straight-forward and can be found in the majority of ELSP papers. Only one product can be produced on the machine at a time. No production can happen in parallel and there are no by-products or cogeneration. While such features might be relevant in practice, they would be too specific to include in a general model.

Point two describes production costs. The model assumes that producing 5 pieces costs half as much as producing 10 pieces. For situations with batch production, the batch size of a production run could be set accordingly. Currently the model doesn't account for stochastic production times, but if necessary it could be adapted for it.

Assumption number three is concerned with production load. Since this paper's model doesn't presuppose any information about contribution margin, an efficient production scheduling when load is higher than 100% is not possible. In order to do this, sales margins need to be taken into account in addition to production costs.

5.1.2 Inventory

The next assumption can be satisfied on similar grounds. While linear holding costs are unrealistic in practice, any other assumption would make the model too specific for general use. If there should be an application that requires a different inventory cost function, like step-fixed costs, this could easily incorporated into the model implementation.

5.1.3 Setup

The second last point deals with setup costs, depending on the predecessor product. Such setup costs are quite frequent in practice (Allahverdi et al., 1999, p. 219) and their deterministic version can be easily represented in a setup time matrix. For the moment, this paper doesn't account for stochastic setup times, but they too could be modeled and optimized within the existing framework.

5.1.4 Demand

The final assumption describes demand modeling. While demand in early ELSP models was deterministic, or transformed into one by bundling orders that arrive within a certain time span, this model assumes deterministically distributed demand. Demand in the current version is based on a compound (stuttering) poisson distribution, i.e. demand *time* and *quantity* are modeled separately. This allows a demand distribution to be fitted to a wide variety of demand series found in practice, including series that are too arbitrary for most standard distributions.

5.2 Model description

The first section of this chapter evaluated the assumptions, this model supposes. The current section builds on these assumptions and describes how they were reproduced in a model.

The first part will deal with the model's global architecture and building blocks. The second part evaluates its core: the transition function, which is modeled as SMDP

5.2.1 Implementation

The model is implemented in object-oriented Java. It consists of a number of classes that interact through interfaces and functions. This design has the big benefit that individual parts can be optimized and interchanged. E.g. a common cycle policy can easily be interchanged for a fixed cycle policy. Or a GA TSP sequencer can be interchanged for closest neighbor.

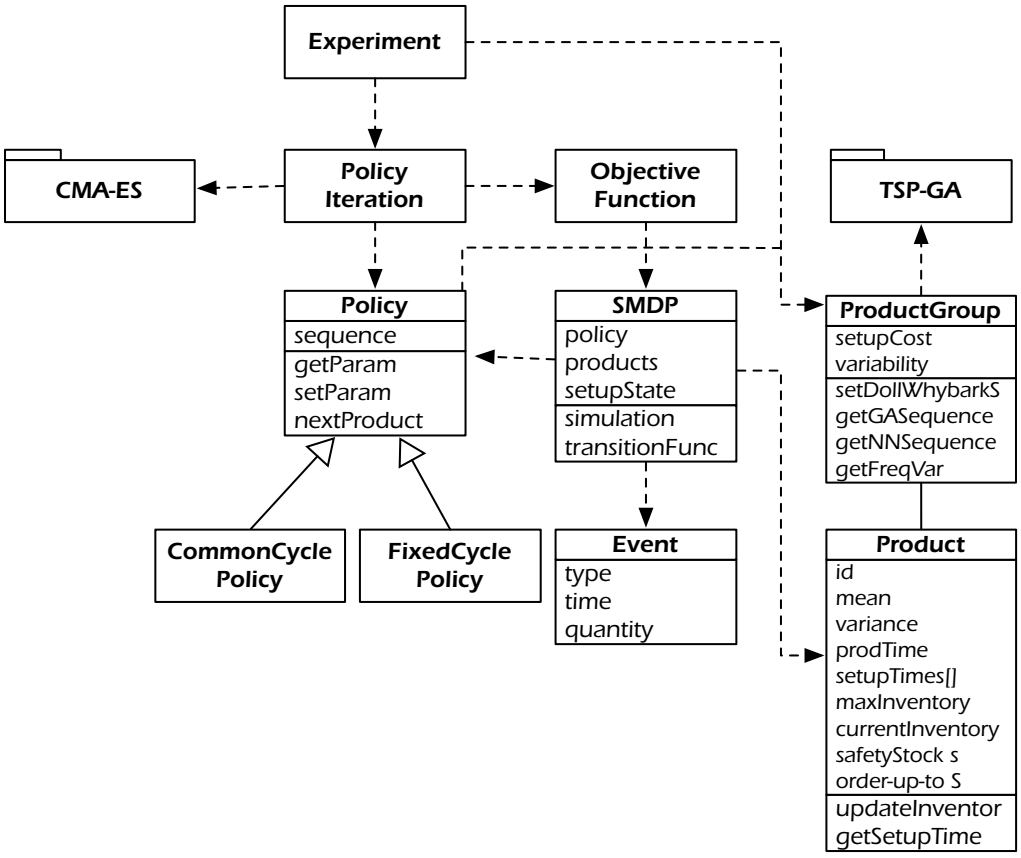


Figure 5.1: *Class diagram of simulation- and optimization model*

Figure 5.1 shows a simplified UML class diagram with the model's main components. At its core there is the class *SMDP*. It evaluates policies and uses information from instances of the *Product*-class to calculate rewards and time events. Demand is modeled in the class *Event*. At each call an *Event* instance is generated, which consists of a product type, demand time and demand quantity.

Policy is a super-class that can be extended by a number of sub-classes. The two examples shown in figure 5.1 are a common cycle- and fixed cycle policy. *Policy* instances pull they information they require about *Product* from *ProductGroup*. This includes calculating aggregate setup cost and sequence variability, or creating a sequence by using the attached TSP GA optimizer[1].

[1]The Java source code for this genetic optimizer for TSP-problems is licensed under the GNU GPL and available from `http://code.google.com/p/java-traveling-salesman/`

The *Experiment* class represents the interface for the end user. It might contain a simulation study or a simple optimization task for a real production facility. *Experiment* uses a specific policy and sets up *Product* via *ProductGroup*. Subsequently the CMA-ES from section 4.1 is used to adapt *Policy* parameters.

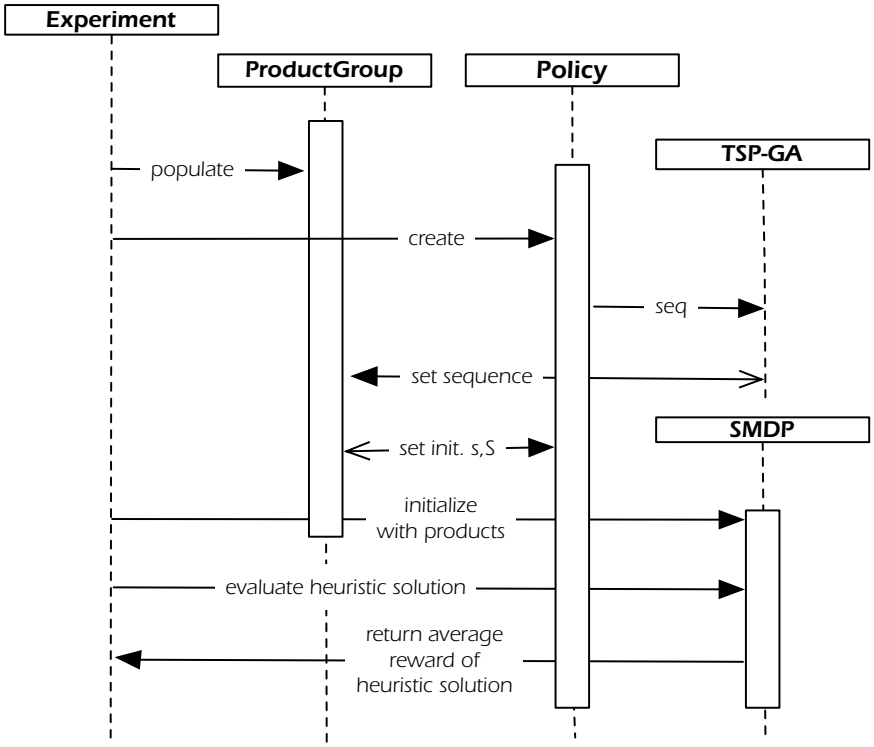


Figure 5.2: Sequence diagram of the initialization phase

When an experiment is started, the first step is to populate a *ProductGroup* with products and their attributes, like mean demand, setup cost, production time and maximum inventory. Next a new *Policy* instance is created. During initialization, the policy uses the *TSP-GA* optimizer to find a cost-minimizing production sequence, which is saved as a production sequence with the product group. Then the heuristic from Doll and Whybark (1973) is used to calculate rough base stock levels.

Once the policy is initialized, the *SMDP* simulation module and *CMA-ES*[2] are used to improve the initial heuristic solution.

The process is started by *PolicyIteration*. The class first initializes the *CMA-ES*, who reads the existing *Policy* parameter to create new points to probe. Subsequently, another class called *ObjectiveFunction* is used as an intermediary to simulate a series of production runs on the *SMDP*. *SMDP* uses *Policy* as a controller to decide on the next product that gets produced, depending on a certain system state. This process is repeated until a certain number of transitions has passed and a good estimation of the current policy parameter's average reward can be made.

[2]The CMA-ES implementation used in this model is available from `http://www.lri.fr/~hansen/cmaes_inmatlab.html`

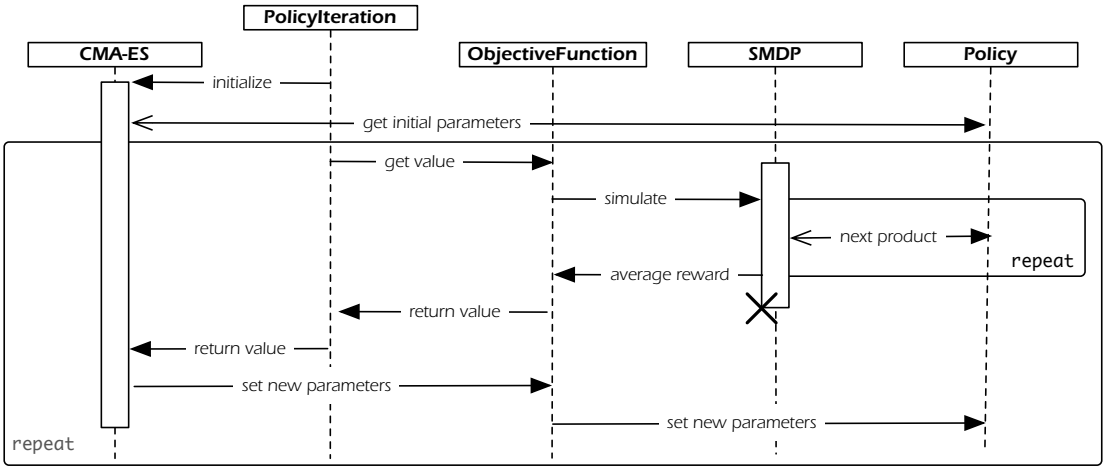


Figure 5.3: Sequence diagram of the optimization phase

This value of the *ObjectiveFunction* is returned to the *CMA-ES*. After internal evaluation of the parameters, it sets new parameters and repeats the evaluation process, after a pre-defined number of enumerations has been reached.

5.2.2 Transition function

The *SMDP* class of the Java model that was explained in the previous section, uses a SMDP to model a production period or demand event. This section gives a rough introduction to SMDP and how they are used to model the production- and decision process presented in this work.

Semi-Markov Decision Process

Markov Decision Processs (MDPs) are a way to model sequential decision that depend on each other and change the state of a system by actions taken by an agent or controller. As shown in figure 5.4, at each stage two things happen. First the decision maker receives a reward. Second, the system moves into the next state. Previous to this transition, a decision or action was taken. Decisions are often based on a policy, who take a specific action based on the current system state. The challenge is to come up with a policy that maximizes reward over a series of states.

As opposed to ordinary MDPs, who follow a discrete time distribution, SMDPs are a more generalized version. They allow a decision maker to choose an action whenever the system state changes. This can also happen at random intervals (like random demand events). The time spent in a particular state is not known in advance. Such models are also referred to as continuous-time models. Their more general form makes them suitable for a wider variety of models than a simple MDP.

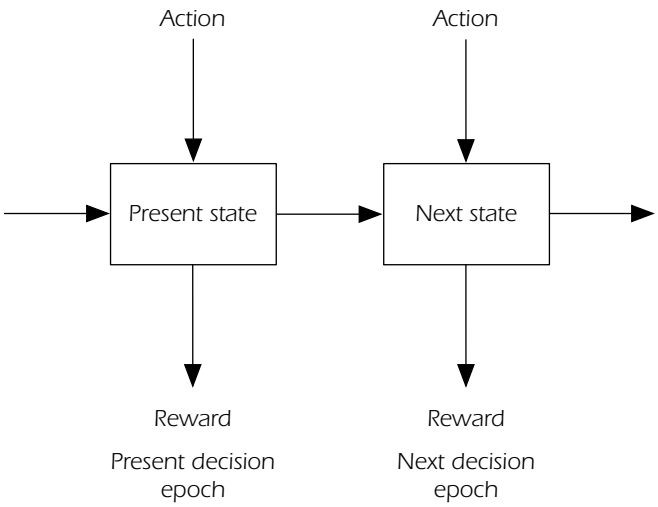


Figure 5.4: Symbolic representation of a sequential decision problem.
(Puterman, 2005, p. 2)

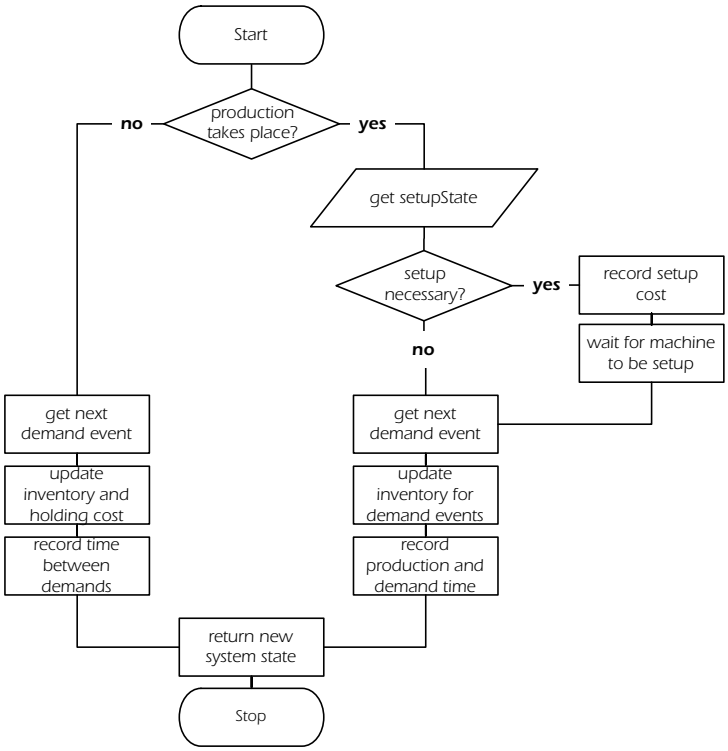


Figure 5.5: Flow chart of transition process in SMDP class.

Model of the transition function

Puterman (2005, p. 531) also notes that in a SMDP, two parallel processes can be distinguished. This comes from the very nature of having random state changes even without actions taken by the decision maker. They refer to the underlying events as *natural process*, the states and decision points relevant for actions are called *SMDP*.

The difference between the natural and SMDP becomes more clear, when taking a look at figure 5.5. Whenever a decision is necessary, the current system state is sent to the policy instance. Depending on the decision rule used, it finds a decision. The transition function gets the next product to manufacture as input.

Initially the system checks, whether actual production is scheduled to take place or not. In cases of high inventory, the policy instance might decide not to produce anything for a period of time. In case production takes place, the setup state needs to be checked. If the machine is already set to the right product, no setup is necessary and production can start. In cases a setup is necessary, the setup time and cost is recorded in the reward array.

The further part of the process can be considered to be natural. A new demand event is generated and the system state changes, i.e. inventory levels are updated. Finishing production runs restock inventory. Then total time and inventory cost are recorded. At this point the system requires another input from the decision policy based on the new state.

5.3 Control policies

Policies are used to combine the different parts that jointly make production decisions. Two main building blocks are required for a control policy. First a production *sequence* needs to be set. Production on the machine happens according to this sequence. Second, the safety- and order-up-to levels need to be set. These might be subject to batch sizes required by the economics of the manufacturing facility.

Based on the properties of the sequence, a major distinction between common- and fixed cycle policies can be made. Common cycle policies only have their sequence set at the beginning of the optimization process. Each product is produced exactly once in a cycle and only stock levels are subject to optimization by the CMA-ES.

The second policy class are fixed cycle policies. These can contain some products more than once and the sequence's parameters are also subject to optimization by the CMA-ES.

The following section describes each policy and how it is different from the others. The integration of a policy in the optimization procedure is elaborated in figure 5.3. The next sections will focus on the inner workings of the policies and differences between them.

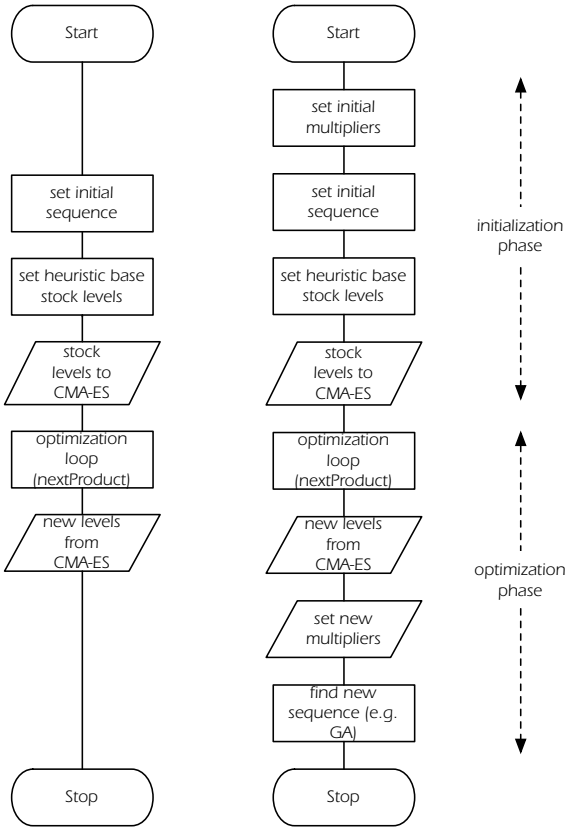


Figure 5.6: Comparison of optimization process within a common cycle (left) and fixed cycle (right) policy.

5.3.1 Common cycle policies

This policy class is based on fixed production cycles with all product multipliers $R = 1$, i.e. every product is manufactured once in a cycle. This makes these policies relatively simple, because only safety stock- and order-up-to levels need to be established. On the downside they perform worse on heterogeneous product mixes.

The strict nature of common cycle policies limits the optimization methods that can be applied to them, when compared to a fixed cycle. Therefor the main focus is on improved sequencing. Here a simple closest-neighbor algorithm is compared to a genetic sequence. This optimization might be based on setup time *or* setup cost. This would mainly depend on the particular application environment. The current work is focused on setup times.

The left part of figure 5.6 illustrates such a policy. After initial base stock levels have been determined, the policy supplies production decisions to the transition function, as described in figure 5.3. After each optimization run, new base stock levels are set, whereas the sequence stays the same. The process of finding a suitable sequence can be summarized as follows:

1. Receive preliminary, sorted sequence S.
2. Optimize sequence S to get final sequence Q with minimized setup cost.
3. Start production with first product in queue Q at position $j = 0$.

Note that the sequence remains untouched after the initial optimization. The CMA-ES-optimizer only tests base stock- and safety levels. The differentiation between policies is made in the second step. The two policies that were tested use a closts-neighbor and a genetic sequencing algorithm.

Common cycle with closest neighbor sequence (CN)

This is the simplest and most naïve policy available. Each product appears exactly once in the cycle. The production sequence is set at the beginning by using the closes neighbor heuristic introduced in section 4.3. Once the sequence has been determined, different base stock levels are tried- and evaluated by CMA-ES.

The *nextProduct*-function uses the product sequence and reorder points to make production decisions for the transition function. This process can be found in (5.3.1).

$$\pi(s;S) = \begin{cases} Q_j & \text{if } y_j < S_j \\ Q_{j+1} & \text{else if } y_{j+1} < s_{j+1} \\ 0 & \text{else} \end{cases} \tag{5.3.1}$$

First the inventory level y_j of the current product in the queue Q at position j is determined. If it has not reached its order-up-to level S_j yet, production continues. In case the current product is finished and the next product in the queue is below its reorder level s_j, production of this product will start. If it should be above its reorder point, the facility goes idle.

Common cycle with genetic sequence (CG)

This policy is very similar to the previous one, in that it sets a sequence once at the beginning and then only evaluates base stock levels. The sequence does not follow the closest neighbor heuristic, but tries to find the best production order globally by using a genetic algorithm. This should reduce setup times by at least 25%. (Matai et al., 2011)

Since setup costs are omitted in this paper, only setup times are evaluated. In production environments with more dominant setup costs, this might be different. Without setup costs, a better production cycle should be able to reduce holding costs. Some of the savings will come from reduced cycle stock, others from reduced safety stock.

The process of deciding on the next product is analogous to (5.3.1).

5.3.2 Fixed cycle policies

As opposed to common cycle policies, which have all products on the same cycle, this kind of policy allows for different cycles. By doing so, products with low demand can have longer cycles and products with high demand are produced more often to keep holding cost low.

The optimization algorithm does not only change base stock levels, but also tests different multipliers for products. The actual procedure works like this:

1. Receive product multipliers R from CMA-ES.
2. Use multipliers R and products P to get preliminary sequence S.
3. Optimize sequence S to get final sequence Q with minimized setup- or holding costs.
4. Start production with first product in queue Q at position $j = 0$.

Using an example with three products, this series of steps can be illustrated with a numerical example. First the multipliers are received either from CMA-ES or the initial heuristic solution.

$$R = \{3, 1, 2\} \tag{5.3.2}$$

Next the multipliers are used to expand the products in a preliminary sequence.

$$S = \{0, 0, 0, 1, 2, 2\} \tag{5.3.3}$$

The final step will sort the sequence in an optimized queue.

$$Q = \{0, 2, 0, 1, 0, 2\} \tag{5.3.4}$$

The first two steps are the same for each policy with a fixed cycle. The differentiation of the individual policies happens in the third step, which basically offers two optimization objectives. There can either be a focus on setup times or holding costs.

In case the former objective is chosen, the optimization algorithm tries to lower setup cost without regard to holding cost. This has a number of advantages and disadvantages. On the upside the time spend with setups over the full cycle, is minimized and therefor more time can be spent on the actual production process. On the other hand, it is also possible that the nature of the setup time matrix favors a clustering of one or more products at the beginning or the end of a sequence. In such a case setup times might be slightly shorter than otherwise, but will be paid for in extra holding cost. This is because in an unbalanced sequence, base stock levels need to be higher than in a balanced one. When very few production runs at one point in the sequence need to satisfy demand for the whole cycle, holding cost are naturally higher. This can outweigh the savings from faster setup times.

For policies that opt for a more balanced sequence and therefor lower base stock levels, similar arguments apply in an inverse way. Most importantly, a balanced cycle allows for lower safety levels because the gaps between production runs are smaller. This will result in lower holding cost, but does not take into account the dependencies caused by sequence-depenent setup cost.

To avoid the problems encountered with the pure variants of the two sequencing options, a mixed multi-objective policy is proposed as an alternative.

Fixed cycle policy with minimized holding cost (FH)

This control policy also uses a genetic algorithm to improve the production sequence. As opposed to the previous policy, the objective are not low setup times, but a balanced sequence. This means that products are spread evenly over the whole cycle and don't cluster at the beginning or end. By taking this into account, overall inventory can be lower.

To determine how "balanced", i.e. evenly spread a production sequence is, a production sequence is converted into frequencies. E.g. sequence $Q = \{2, 1, 3, 4, 5, 3, 2, 3, 5, 2, 5, 1\}$ becomes the following frequency, when only considering product $F_3 = [2, 1, 6]$. This would put the average distance between product 3 in the sequence at $\frac{2+1+6}{3} = 3$.

Next the variance of frequencies is calculated by summing over the difference between the average- and individual distances. $\text{var} = (2-3)^2 + (1-3)^2 + (6-3)^2 = 14$. This example results in a rather high frequency variance for product 3, because it only gets produced in the

beginning of the sequence. The last batch would need to be higher to cover for the extended period between the first and last production run.

The right part of figure 5.6 displays a generic fixed cycle policy. In addition to setting heuristic base stock levels, the policy also needs to determine multipliers for each product. These are then used to construct a sequence and optimizing it by keeping the frequency variability as low as possible.

After each optimization run, new multipliers are set and a new sequence is constructed. This makes a fixed cycle policy computationally more expensive than a common cycle policy.

Fixed cycle policy with minimized setup cost (FS)

In the previous policy, the pure variant of a balanced sequence with holding cost as an indirect objective was used. The current policy implements a pure form of a minimized setup cycle. The genetic optimizer has the objective to achieve the shortest possible setup times.

At the same time, it's not allowed to have the same product consecutively in the cycle. For such a case setup times are set to an arbitrary high value and the CMA-ES should reduce the multipliers R if it can not fit all the products within the cycle in a sensible way.

Fixed cycle with multi-objective sequence (FM)

This production policy is mixture of the two previous ones. The main difference lies in the objective function of the sequence optimization. The target function is multi-objective and uses a weight based genetic algorithm, as described in section 4.2.4. The first objective of achieving a balanced cycle to save inventory cost stays the same. In addition, the GA also attempts to lower setup times over a full production cycle.

The importance (weight) of a short, as opposed to a balanced cycle is determined by CMA-ES. Over time the optimizer can find the best combination of production sequence- and base stock parameters.

5.3.3 Preemptive policies

In previous research projects by Graves (1980) and Löhndorf and Minner (2011), the authors suggest a preemption-rule for "emergencies" with low inventory levels. In situations without sequence-dependent setup times, this enhancement lowered average cost in all cases. This

consideration could be accounted for by an extension of the formula introduced in (5.3.1).

$$\pi(s;S) = \begin{cases} Q_j & \text{if } y_j < S_j \\ Q_{j+1} & \text{else if } y_{j+1} < s_{j+1} \\ Q_{j_n^+} & \text{else if } \exists \text{ n: } y_n \leq s'_n \\ 0 & \text{else} \end{cases} \tag{5.3.5}$$

where $Q_{j_n^+}$ is defined as the next occurance of product n in queue Q.

The policy also needs to introduce a new stock level s'. Löhndorf and Minner (2011) call this variable preemption point. In contrast, s as was used in previous policies can be described as can-order point. In normal operation the policy cycles along the fixed cycle product queue. Products below their can-order level s are produced up to their target level S. After each production run, the policy also checks for products below their preemption point. If it finds them, production jumps to the next occurence of the affected product in the cycle.

When applying such an additional rule to SDS, the policy has to weigh the cost of leaving the efficient setup time cycle against possible lost-sales cost. Excess setup times arise at the point of preemption and when reentering the cycle. For a preemption to actually *lower* reward, the extra holding cost caused by these two steps need to be balanced against average expected demand events in the same time span.

6 Numerical comparison of policies

This final chapter presents the results of the simulation study that was conducted to compare different performance aspects of the production scheduling policies introduced earlier. The experimental setup and generation of sample data is based on Löhndorf and Minner (2011) with minor modifications to account for sequence-dependent setup times.

6.1 Experimental design

To make substantiated claims on the performance and influence of different control policies, they need to be tested on a larger number of sample instances. To do so, a number of design parameters were introduced. Then a Sobol sequence was used to create different product parameters for sample runs. Since the number of products determines the complexity of the problem, samples were generated for groups of 3, 5, 10 and 15 products.

6.1.1 Design parameters

To have more control over sample properties, design parameters are used. They set the upper- and lower bounds of sample instances. By tweaking them, the model can be tested to account for various aspects, like load factor or demand variability.

The intervals used to sample the product parameters are displayed in table 6.1. *Avg Mean Demand* per period describes the average demand for a product, *CV demand* represents the degree of demand variability. Higher variability means less predicatiblity because rare, large orders can have a big impact on inventory. With increasing specialization and diversification happening in many sectors, variability for single products is expected to increase, while selling a small number of standardized products should reduce demand variability.

Lost Sales Cost are used as a "punishment" for stockout situations. In practice this value is difficult to measure. It doesn't only include contribution margins lost on sales, but also intrinsic losses, like a decrease in customer satisfaction and trust.

The *Avg Holding Cost* parameter governs the cost of inventory that needs to be paid per item on stock. In general it includes a wide range of cost, a company can be face. These include capital expenditures, depletion, shrinkage, storage and risk of price fluctuations. To avoid

excessive holding cost, one aim of the optimization procedure is to keep inventory levels as low as possible, while still avoiding stockout situations.

Load Factor is used to derive production times. This parameter needs to be < 1. Otherwise demand can't be met any more and the more profitable products need to be prioritized. This is beyond the scope of traditional production planning and shouldn't occur on a regular basis in practice. In the case of constant shortages and lost sales of products with positive contribution margin, any rational firm would either increase the sales price of their product or invest in additional capacity.

Avg Setup/Prod Time gives the time necessary for production of items and setups between production runs. Production time is assumed to be linear, i.e. a higher quantity means a longer production time. Since this work deals with SDS, simple average setup times are not enough. A whole setup time matrix is required to govern relationships between different products. For this reason, an extra parameter, *Avg CV Setup Times* was introduced. This design parameter was not used in Löhndorf and Minner (2011) and describes the coefficient of variance of the setup time matrix for each product. First average setup times for each product is determined in the same way as in the reference paper mentioned before. Subsequently the setup times for each product are expanded one more time to arrive at cross-setup times for each product. The parameter *Avg CV Setup Times* governs the interval of setup times, with average setup time for each product as base value. Like all other values, the interval is then samples and mixed randomly. The results are written in a setup time matrix.

Design Parameter	Min. Value	Max. Value
Avg Mean Demand	5	25
Avg Lost Sales Cost	100	100
Avg Holding/LS Cost	0.001	0.01
Avg Setup/Prod Time	70	100
Avg CV Demand	0.5	1.5
Div CV Demand	0.3	0.9
Avg Load Factor	0.3	0.6
Avg CV Setup Times	0.0	0.5

Table 6.1: Intervals of design parameters used in the experiment

6.1.2 Product parameters

In order to arrive at definitive sample instances, the design parameters described in section 6.1.1 need to be sampled in some way. This could be achieved by random sampling. To cover the whole sample space more efficiently, a technique calles quasi-Monte Carlo sampling will be used. The aim is to generate a low-discrepancy sample that closely follows a uniform distribution. One way to arrive at such a sample are digital nets/sequences. They use the expansion of $i - 1$ in a certain base. (Lemieux, 2009, p. 139)

This work uses a Sobol sequence with 11 dimensions for each sample instance. The points are calculated by using the Java-based SSJ-library[1]. In total 4 times 1000 instances were generated, with the number of products ranging from 3 to 15.

6.1.3 Simulation runs

Subsequently the five sets of sample instances generated in the previous section were then solved with each production policy and the results recorded. To decrease variability, average rewards are collected using batch means.

The actual processing was done on Virtual Machine (VM) instances in the Amazon Elastic Computing Cloud (EC2). A VM was set up to receive a new task from a central command and control database that contained all the 4 000 problem instances necessary for each of the five policies.

In the next step, the VM was cloned and simultaneously executed on 100 instances with a combined number of 800 CPU cores and 700 GB of memory.

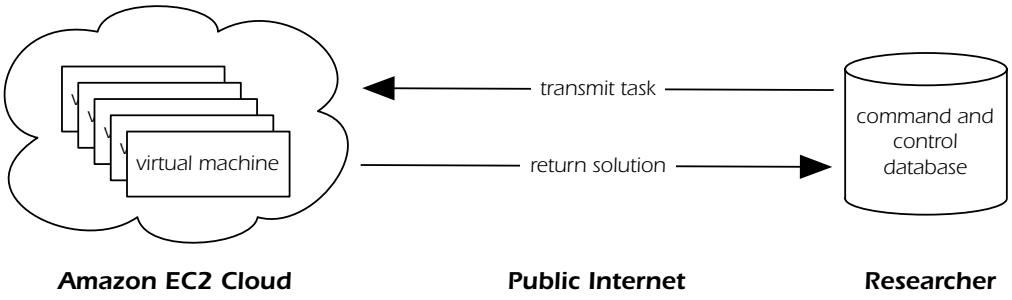


Figure 6.1: Simulation runs in the Amazon cloud, using a control database.

6.2 Analysis of results

By using the data generated from the simulation runs, the performance of control policies, as well as the environment in which they're most effective can be tested.

First the total setup cycle times are of interest. If a simple greedy NN algorithm can find the shortest sequence as well, there would be no need to choose a computationally more expensive GA. After arriving at a lower setup sequence time, this advantage also has to translate in lower average rewards. If lower setup times don't decrease holding cost significantly, they can as well be ignored. It should be noted that this is naturally dependent on the sample data. There will definitely be cases, when setup times are so insignificant that setup cycle optimization does not make a big difference.

[1]The SSJ-library is available from `http://www.iro.umontreal.ca/~simardr/ssj`

Next, the difference between fixed- and common cycle solutions shall be analyzed. A more flexible cycle with some products appearing more often than others should lead to lower overall costs, compared to having all products on the same (common) cycle. By differentiating between sample groups with low- and high demand variability, the cost advantage gained by using a flexible cycle can be observed as well.

In the realm of fixed cycle policies, the performance of the two possible objectives for optimization is of interest. As discussed in section sec:fixedCyclePolicies, the sequence can either be optimized to low holding cost, low setup times, or a mix of both. The numerical study will show which objective yields the best outcome.

6.2.1 Length of common cycle production sequence

Under SDS the total setup times (or cost) are an important aspect. By comparing the length of closest-neighbor and GA sequences, improvements in cycle time can be observed.

N	Size	CN	GA	Δ
3	1000	20.4	5.7	72%
5	"	27.6	10.4	62%
10	"	45.4	23.3	49%
15	"	62.5	37.7	40%
20	"	80.9	53.9	33%

Table 6.2: Comparison of total cycle setup times for different problem sizes.

Neither the average setup time, nor the variance of setup times seem to have a big influence on the performance gap between NN and GA sequence. The only factor that influences the performance gap is the number of products to optimize. The more products there are in the cycle, the closer is the NN to GA. This observation is in line with Matai et al. (2011), who suggest a performance gap of about 25% for a high number of products.

6.2.2 Influence of design parameters on policy performance

Some design parameters have a bigger impact on the final result than others. To make forecasts, about the impact of different settings, coefficients of determination were calculated for each design parameter and policy. The average reward was normalized by dividing through the number of products. This leaves the reward per product as dependent variable in the analysis. Individual design parameters are the independent variables.

The results are similar to those observed by Löhndorf and Minner (2011, p. 21). The highest influence is exerted by the average load factor. In the two common cycle cases, about 40% of the average reward can be explained by changes in the production facility's rate of utilization. In the case of fixed cycle policies, this value decreases to about one third.

	CN		CG		FM		FH		FS	
Factor	r_2	F	r_2	F	r_2	F	r_2	F	r_2	F
Number of Products	0.01	51	0.01	49	0.00	27	0.00	24	0.01	54
Div Mean Demand	0.04	183	0.04	191	0.04	196	0.04	198	0.03	165
Div Lost Sales Cost	0.00	0.1	0.00	0.0	0.00	0.0	0.00	0.0	0.0	0.0
Avg Holding/LS Cost	0.23	1225	0.24	1290	0.34	2135	0.30	1780	0.30	1731
Div Holding/LS Cost	0.00	0.0	0.00	0.0	0.00	0.5	0.00	1.2	0.00	0.1
Avg CV Demand	0.02	106	0.04	144	0.05	214	0.03	118	0.01	58
Div CV Demand	0.00	2	0.00	3	0.00	2	0.00	0.1	0.00	2
Avg Load Factor	0.43	3042	0.40	2726	0.32	1914	0.25	1354	0.35	2159
Div Load Factor	0.00	2	0.00	3	0.00	2	0.00	0.1	0.00	2
Avg Setup/ProdTime	0.00	19	0.00	20	0.00	19	0.00	21	0.01	28
Div Setup/ProdTime	0.00	0.1	0.00	0.6	0.00	0.2	0.00	0.1	0.00	0.8
Linear Model	0.73	420.9	0.73	402.5	0.75	410.0	0.62	317.9	0.71	381.8

Sample Size = 4000, r_2 = coefficient of determination, F = F test statistic.

Table 6.3: Relationship between design parameters and cost per product.

With lost sales cost fixed at 100 the ratio of lost sales to holding cost also seems to have a relatively high influence. The influence is higher for fixed- than for common cycle cases.

These two parameters are the only ones exerting major influence. Löhndorf and Minner (2011) also observed a relatively high influence of the setup- to production time ratio. This was not the case in this work. This is most likely caused by two factors. First the corresponding interval was chosen narrower than in the reference paper to arrive at higher setup times in relation to production time.

Next it should be noted that only a small subset of the actually generated setup times is really used. In the ten-product case, only 10 setup times out of a matrix of 100 are used. Those values are most likely on the lower end, because both, the closest-neighbor and genetic algorithm try to keep setup times low.

Besides the two design parameters displaying major influence, there are the two demand parameters. They seem to have a minor positive correlation on the outcome. Meaning higher average demand or demand variance slightly increase average rewards (cost). Still, this effect is not strong enough to draw any conclusions.

6.2.3 Comparison of policies

The most important benchmark of the performance of a policy is average reward. A lower average reward means less holding- and lost sales cost. At the same time there is a variety of factors that can influence the end result. These influences were established in table tab:relationship.

The most relevant factor in this table was load factor, which is also the first classification made

for comparing different policies.

The next highest influence can be contributed to average holding- and lost sales cost. Using this ratio is problematic, because lost sales cost are mostly used to direct the optimization heuristic. The next best choice is the diversity between mean product demands. This parameter is also relevant to companies seeking to deploy a certain policy.

N	load	cvDemand	Size	BEST	CN	CG	FM	FH	FS
3	low	low	250	0.56	0.65	**0.60**	**0.62**	0.89	0.81
		high	250	0.78	0.85	**0.83**	**0.89**	1.08	1.00
	high	low	250	1.10	1.76	1.47	**1.16**	1.47	1.42
		high	250	1.38	1.98	1.69	**1.43**	1.68	1.64
5	low	low	250	0.55	0.67	**0.62**	**0.57**	0.78	0.82
		high	250	0.79	0.93	**0.88**	**0.82**	0.95	0.95
	high	low	250	1.08	1.67	1.50	**1.12**	1.40	1.54
		high	250	1.35	1.99	1.81	**1.38**	1.58	1.64
10	low	low	250	0.57	0.73	0.65	**0.59**	0.63	0.81
		high	250	0.81	1.03	0.94	**0.82**	0.87	0.97
	high	low	250	1.19	1.74	1.51	**1.20**	1.27	1.69
		high	250	1.46	2.10	1.91	**1.47**	1.53	1.88
15	low	low	250	0.61	0.80	0.70	**0.65**	**0.64**	0.87
		high	250	0.85	1.11	1.01	**0.90**	**0.88**	1.03
	high	low	250	1.30	2.07	1.81	**1.40**	**1.35**	1.84
		high	250	1.55	2.45	2.17	**1.63**	**1.62**	2.03
Mean			5000	1.00	1.41	1.26	**1.04**	1.16	1.31

Table 6.4: Normalized average reward for each product policy.

Table 6.4 is split by these two design parameters, as well as the number of products. The values displayed are the average observed rewards over a sample of 1000 in total. Because the samples are split based on design parameters, every subgroup comprises a sample of 250 cases. In addition to the four policies that were tested, an extra pseudo-group comprising the best value observed over all available results was calculated as well.

As expected, the biggest impact is caused by the load factor. The sample with higher load factor has about 70% higher cost than the lower half when taking all four policies into account. Higher load factors are a general challenge for production planning. They require more precise planning and less room for errors.

When comparing the five policies over all sample groups and numbers of products, the fixed cycle policy with GA-optimized sequence that takes setup cost *and* a sequence variability into account finishes first. On average it's only 5% worse than the corresponding best result observed.

The difference in setup cycle times has already been elaborated in table 6.2. When comparing total rewards, the impact of a shorter setup cycle on average rewards can be observed as well. As expected the genetically optimized sequence is always better than the closes-neighbor version. The difference is most significant at a high load factor.

			CN		CG		FM		FH		FS	
N	load	cvDemand	MAD	frq	MAD	frq	MAD	frq	MAD	frq	MAD	frq
3	low	low	0.13	26%	**0.08**	**45%**	0.08	17%	0.38	12%	0.26	5%
		high	0.13	40%	**0.09**	**42%**	0.13	13%	0.33	6%	0.23	5%
	high	low	0.69	3%	0.45	16%	**0.14**	**61%**	0.46	19%	0.37	15%
		high	0.65	5%	0.41	22%	**0.14**	**53%**	0.39	21%	0.3	17%
5	low	low	0.14	7%	0.09	12%	**0.08**	**67%**	0.28	14%	0.29	8%
		high	0.16	10%	0.11	17%	**0.07**	**60%**	0.19	12%	0.18	18%
	high	low	0.6	1%	0.43	1%	**0.15**	**80%**	0.39	18%	0.48	7%
		high	0.67	1%	0.48	0%	**0.13**	**77%**	0.3	22%	0.35	18%
10	low	low	0.18	2%	0.11	16%	**0.07**	**50%**	0.1	31%	0.25	2%
		high	0.24	1%	0.16	7%	**0.05**	**62%**	0.1	30%	0.19	10%
	high	low	0.59	0%	0.39	8%	**0.12**	**58%**	0.17	34%	0.54	1%
		high	0.69	0%	0.5	3%	**0.11**	**60%**	0.16	37%	0.45	3%
15	low	low	0.2	2%	0.13	21%	0.07	32%	**0.05**	**46%**	0.26	1%
		high	0.26	0%	0.18	9%	0.08	37%	**0.06**	**53%**	0.19	7%
	high	low	0.79	0%	0.57	9%	0.15	32%	**0.15**	**60%**	0.55	2%
		high	0.91	0%	0.64	2%	0.16	43%	**0.15**	**54%**	0.48	3%
Mean			0.43	6%	0.30	15%	**0.10**	**49%**	0.22	30%	0.33	7%

Table 6.5: Frequency of choosing the best policy and corresponding MAD.

When comparing the average advantage gained by using a flexible cycle with different product frequencies, the advantage is slightly higher with more heterogeneous demand. (0.19 to 0.23).

Another interesting result is the comparison between the three fixed cycle policies FM, FH and FS. The latter two use just one objective for optimization (setup times or holding cost). The first one is multi-objective and the CMA-ES decides on the specific weight of the two objectives. In the numerical study, the mixed policy FM yielded the best results and was on average only 5% worse than the best result observed.

FH, which focuses on holding cost, performed rather poorly for 3 and 5 products. At 10 and 15 products, the policy closed up to FM and clearly outperforms FS

The fixed cycle policy with a pure focus on setup times (FS) performs worst of all fixed cycle policies. This means that minimizing setup times is not sufficient in fixed cycle sequences. There has to be an additional focus on a balanced cycle, i.e. low holding cost. Especially when sequences get longer, a badly balanced cycle can initiate the CMA-ES to increase base stock

levels too much. One explanation for this behavior might be holding cost becoming more relevant in comparison to lower setup times (which cause holding cost as well).

It can also be assumed, that CMA-ES is setting the α-parameter higher, when encountering a higher number of products. Meaning it gives a higher priority to holding- than setup cost. Otherwise the results wouldn't be so close. Especially when comparing FH and FS.

The final table complements 6.4. It shows the normalized Mean Average Deviation (MAD) from the corresponding best policy in the sample class, as well as the frequency of how often the policy was equal to the best observed reward. All in all the findings in table 6.4 are confirmed. The FM policy gives the best result for about half the time. When omitting the sample instance with 15 products, the number increases to 54%. For all instances it is 49%.

The CN-policy performs worst and only corresponds with the best result in 6% of the cases. Most of these occur in the two 3-product, low load-factor cases. With only three products to choose from, there is a relatively good chance for the NN to pick the same optimal sequence as the GA. With a rising number of products, this probability gets smaller.

The next CG-policy performs rather well in the first two sample instances and even beats the FG in these two cases. The gap is not significant, as can be seen from the tiny MADs. The differences to FM stay small in the low load-factor cases, but are more significant with higher load factors. In total the CG-policy equals the best result in 15% of the cases. In comparison to the CN-policy the MAD rises in a similar, but slightly lower way.

Next in line is the FM-policy. It equals the best policy around half the time. It also features the lowest MAD, meaning that when it is not the best policy, the difference is very minor and insignificant in most cases. The next-best fixed cycle policy FH delivers the best policy in about a third of the sample cases. This is mainly due to its good performance in the high 15-products case. In the high-load samples with 15 products, FH finds the best policy in 60% and 54% of the time. This is about 10% better than FM. The MAD from the best solution is about the same, meaning that the difference between the two policies is rather small.

All in all the numerical study presents the FM-policy with multiple objectives as clear recommendation for up to 10 products. When the policy is used for more than 15 or 20 products, holding cost should be compared to the importance of SDS. For a small number of similar products (or maybe product families) a simple common cycle CG also works.

7 Summary and outlook

7.1 Summary

7.1.1 Literature review

This work started out with an overview of existing research in ELSP. Conditions of the respective solution method were used to guide through the process. While the first attempts to solve ELSP-problems relied on very restrictive conditions, like strictly deterministic demand rates and sequence-independent setup times, these were later relaxed to give rise to more complicated models.

Next the notations for solving common cycle models were introduced. By using the formulas found in Elmaghraby (1978), it was shown that deterministic ELSP without SDS can be solved by using one common cycle length for all products and balancing the length against the holding- and setup cost. This is similar calculating optimal order quantities by using the well-known EOQ-formula.

A common drawback of the common cycle method is its inflexibility when demand rates between products are very heterogeneous. In many practical cases, a small percentage of products accounts for the largest share in revenue and sales volume. By contrast, many other products are sold less often. It would be more beneficial for companies to produce high-volume products more often.

To solve this issue, researchers introduced basic periods. The common cycle model was extended by integer multiples and a basic period term. This made it possible to differentiate between high- and low volume products. On the downside computational complexity increased and the research community resorted to DP and metaheuristics.

Despite some improvement, the basic period approach still had some drawbacks. Namely the need equal production sizes in all runs. This led to the introduction of time-varying lot sizes. Instead of integer multiples, this method features a whole production sequence. Lot sizes are no longer equal, but vary according to the time between production runs. This approach is still the most popular, but is NP-hard and requires the use of a search heuristic.

The succeeding section of the literature review introduces sequence-dependent setup times to deterministic ELSP. While earlier researchers, like Maxwell (1964) tried to work out special

cases in the setup time (or cost) matrix, most researchers moved on to apply TSP techniques to the problem. SD-ELSP literature has closely tracked this related research class ever since.

To solve TSP, three kinds of solution approaches are available. Exact methods, like branch-and-bound or dynamic programming guarantee the best solution, but are infeasible for larger problems. Heuristic- and metaheuristic techniques can find a "good" solution faster. Some popular metaheuristics that have been applied to the problem include GRASP, simulated annealing and a variety of genetic algorithms.

Then the literature available on the stochastic version of the problem was summarized. While the preceding problem classes could still be solved to optimality, stochastic problems require a different approach. First the problem of product sequencing needs to be solved, i.e. when to produce what. This could either happen by cyclic or dynamic sequencing. The former is easier to simulate and optimize. The latter should theoretically offer a better sequence, but also requires constant decision-making by a set of rules or an AI.

The chapter is concluded by a section on sequence-dependent stochastic ELSP. For the lack of specified research, only two case studies that deal with the problem are cited. Those are explained in greater detail in a succeeding chapter.

7.1.2 Problem description

The next part specifies the problem and introduces two practical applications of it. The first of them deals with a Dutch pharmaceutical company, wishing to consolidate their inventory holdings from three to one location. While the actual produce sold in these three markets is the same, packaging represents a bottleneck. Currently the firm artificially creates a deterministic production problem, by planning in cycles. With the added sales data and flexibility, a consolidated inventory would require stochastic planning. Another possible application comes from a pre-Deco company. Similar to the first case study, the company also wishes to reduce inventory by consolidating it. Secondly production costs are unnecessarily high because of suboptimal sequencing.

Then the critical elements of a production plan were elaborated. First a sequence needs to be determined. This might be done by using a fixed- or common cycle. The key difference is that a fixed cycle can contain a product more than once. This gives added flexibility for products with heterogeneous demand.

In addition to a sequence, lot sizes are required as well. When facing deterministic demand, lot sizes can be derived from the sequence. In a stochastic environment, frameworks are used to set inventory boundaries.

7.1.3 Solution methods

In the fourth chapter, some tools and methods for solving the SD-SELSP were introduced. One of them is the CMA-ES. It uses a multi-variate normal distribution to sample a solution space and generate new solution parameters. A genetic algorithm is used to solve the sequencing problem. GAs are modeled from natural selection and work on the basis of chromosomes and fitness functions. The former encode possible solutions. The latter represent the objective function. Mutation and combination are used to generate new candidate solutions. In addition to a GA, a closes neighbor heuristic is introduced to provide a benchmark.

7.1.4 Model and control policies

The fifth chapter introduces the model and suggested control policies. First the model's assumption, like production, inventory, setup and demand behavior are clarified. Then the high-level implementation of the Java simulation-model, as well as the interaction of individual components with each other are described. In terms of control policies, two common- and two fixed cycle implementations are suggested.

7.1.5 Numerical comparison of policies

The final chapter made a numerical comparison of the four control policies. Each of them was tested on 1000 problem instances, which were generated by using a number of design parameters. Then results were analyzed in terms of setup cycle time and average rewards. The results confirmed the improved cycle times of the GA, as well as better results for a more flexible sequence, instead of a common cycle policy. Sample instances with more heterogeneous product demands benefit more from added flexibility than those with less variability.

On the comparison between the two possible optimization objectives in a fixed cycle sequcene, a pure focus on setup times does not seem to be too benefitial. It is better to take both criteria into account and let the CMA-ES decide on the right weights.

7.2 Outlook and further research opportunity

This work can be viewed as a first proposal on how to solve the class of SD-SELSPs, which is harder than it seems at first. Despite the big volume of research done in ELSPs, only some of the concepts can be applied to this problem. Solutions need to be found for the sequencing, as well as the lot sizing problem. Methods that can solve both of them at the same time probably exist as well.

A weakness of the current model is the lack of a preemption policy, when a product runs low on inventory outside its usual production sequence. Both Graves (1980) and Löhndorf and Minner

(2011) find their respective preemption policy to outperform other policies. With sequence-dependent setup times, it is not as simple. Whenever a low inventory situation occurs, the optimal sequence has to be left. This can cause higher-than-necessary setup time at the initial switch. This can result in another shortage, before production of the preempted product is over. Subsequently the optimal sequence is left again, amplifying the effect even further.

The solution might be a more sophisticated preemption rule that takes potential lost sales cost and extra holding cost for increased setup times into account. That way the number of preemptions can be reduced. In addition, any such policy needs to ensure that preemptions are the exception, not the rule. Maybe ADP can be used for this purpose. Another possibility would be a preemption-threshold that is set by CMA-ES.

An additional research direction might be the adaption of a version of ADP for SD-SELSP. This policy was also proposed by Löhndorf and Minner (2011), but did not perform well for sample cases with more than 3 products.

Bibliography

Allahverdi, A., Gupta, J., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, *27*(2), 219–239.

Ashayeri, J., Heuts, R., Lansdaal, H., & Strijbosch, L. (2006). Cyclic production-inventory planning and control in the pre-deco industry: A case study. *International Journal of Production Economics*, *103*(2), 715–725.

Axsater, S. (2006). *Inventory control*. Springer Science+ Business Media, Inc, 233 Spring Street, New York, NY, 10013, USA,.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. New York: Oxford University Press.

Barnes, J., & Vanston, L. (1981). Scheduling jobs with linear delay penalties and sequence dependent setup costs. *Operations Research*, 146–160.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control*. Belmont, Mass.: Athena Scientific.

Bomberger, E. (1966). A dynamic programming approach to a lot size scheduling problem. *Management Science*, 778–784.

Bourland, K., & Yano, C. (1994). The strategic use of capacity slack in the economic lot scheduling problem with random demand. *Management Science*, 1690–1704.

Buzacott, J., & Dutta, S. (1971). Sequencing many jobs on a multi-purpose facility. *Naval Research Logistics Quarterly*, *18*(1), 75–82.

Delporte, C., & Thomas, L. (1977). Lot sizing and sequencing for n products on one facility. *Management Science*, 1070–1079.

Dobson, G. (1987). The economic lot-scheduling problem: achieving feasibility using time-varying lot sizes. *Operations Research*, 764–771.

Doll, C., & Whybark, D. (1973). An iterative procedure for the single-machine multi-product lot scheduling problem. *Management Science*, 50–55.

Elmaghraby, S. (1978). The economic lot scheduling problem (elsp): review and extensions. *Management Science*, 587–598.

Feo, T., Sarathy, K., & McGahan, J. (1996). A grasp for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers & Operations Research*, *23*(9), 881–895.

Gallego, G. (1990). Scheduling the production of several items with random demands in a single facility. *Management Science*, 1579–1592.

Ghosh, A., & Dehuri, S. (2004). Evolutionary algorithms for multi-criterion optimization: A survey. *International Journal of Computing & Information Sciences*, *2*(1), 38–57.

Gilmore, P., & Gomory, R. (1964). Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations Research*, 655–679.

Graves, S. (1980). The multi-product production cycling problem. *AIIE Transactions*, *12*(3), 233–240.

Hansen, N. (2006). The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, 75–102.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.

Harris, F. (1913). How many parts to make at once, factory mag. *Manage*, *10*(2), 135–136.

Hsu, W. (1983). On the general feasibility test of scheduling lot sizes for several products on one machine. *Management Science*, 93–105.

Huang, J., & Yao, M. (2008). A genetic algorithm for solving the economic lot scheduling problem in flow shops. *International Journal of Production Research*, *46*(14), 3737–3761.

Khouja, M., Michalewicz, Z., & Wilmot, M. (1998). The use of genetic algorithms to solve the economic lot size scheduling problem. *European Journal of Operational Research*, *110*(3), 509–524.

Kim, D., Kim, K., Jang, W., & Frank Chen, F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, *18*(3-4), 223–231.

Leachman, R., & Gascon, A. (1988). A heuristic scheduling policy for multi-item, single-machine production systems with time-varying, stochastic demands. *Management Science*, 377–390.

Leachman, R., Xiong, Z., Gascon, A., & Park, K. (1991). Note: An improvement to the dynamic cycle lengths heuristic for scheduling the multi-item, single-machine. *Management science*, 1201–1205.

Lemieux, C. (2009). *Monte carlo and quasi-monte carlo sampling*. New York: Springer.

Löhndorf, N., & Minner, S. (2011). Simulation optimization for the stochastic economic lot scheduling problem. *Working Paper*.

Matai, R., Shing, S. P., & Mittal, M. L. (2011). Traveling salesman problem: An overview of applications, formulations, and solution approaches. In D. Davendra (Ed.), *Traveling salesman problem, theory and applications* (chap. 1). Rijeka: InTech.

Maxwell, W. (1964). The scheduling of economic lot sizes. *Naval Research Logistics Quarterly*, *11*(2), 89–124.

Merz, P., & Freisleben, B. (1997). Genetic local search for the tsp: New results. In *Proceedings of 1997 ieee international conference on evolutionary computation (icec'97): April 13-16, 1997, university place hotel, indianapolis, in usa* (p. 159).

Miller, D., Chen, H., Matson, J., & Liu, Q. (1999). A hybrid genetic algorithm for the single

machine scheduling problem. *Journal of Heuristics*, *5*(4), 437–454.

Moon, I., Silver, E., & Choi, S. (2002). Hybrid genetic algorithm for the economic lot-scheduling problem. *International Journal of Production Research*, *40*(4), 809–824.

Paternina-Arboleda, C., & Das, T. (2005). A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simulation Modelling Practice and Theory*, *13*(5), 389–406.

Pinedo, M. (2008). *Scheduling: theory, algorithms, and systems* (3rd ed ed.). New York: Springer.

Puterman, M. L. (2005). *Markov decision processes: discrete stochastic dynamic programming*. New York: Wiley.

Raza, A., & Akgunduz, A. (2008). A comparative study of heuristic algorithms on economic lot scheduling problem. *Computers & Industrial Engineering*, *55*(1), 94–109.

Raza, S., Akgunduz, A., & Chen, M. (2006). A tabu search algorithm for solving economic lot scheduling problem. *Journal of Heuristics*, *12*(6), 413–426.

Roundy, R. (1989). Rounding off to powers of two in continuous relaxations of capacitated lot sizing problems. *Management Science*, 1433–1442.

Rubin, P., & Ragatz, G. (1995). Scheduling in a sequence dependent setup environment with genetic search. *Computers & Operations Research*, *22*(1), 85–99.

Sengoku, H., & Yoshihara, I. (1998). A fast tsp solver using ga on java. In *Third international symposium on artificial life, and robotics.*

Sox, C., Jackson, P., Bowman, A., & Muckstadt, J. (1999). A review of the stochastic lot scheduling problem. *International Journal of Production Economics*, *62*(3), 181–200.

Sox, C., & Muckstadt, J. (1997). Optimization-based planning for the stochastic lot-scheduling problem. *IIE Transactions*, *29*(5), 349–357.

Strijbosch, L., Heuts, R., & Luijten, M. (2002). Cyclical packaging planning at a pharmaceutical company. *International Journal of Operations & Production Management*, *22*(5), 549–564.

Swamidass, P. (2000). Abc analysis or abc classification. In P. Swamidass (Ed.), *Encyclopedia of production and manufacturing management.* Boston: Kluwer Academic Publishers.

Tan, K., Narasimhan, R., Rubin, P., & Ragatz, G. (2000). A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. *Omega*, *28*(3), 313–326.

Tan, R., et al. (1997). Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. *Omega*, *25*(6), 619–634.

Vergin, R., & Lee, T. (1978). Scheduling rules for the multiple product single machine system with stochastic demand. *Infor*, *16*(1), 64–73.

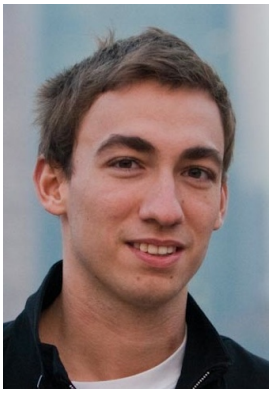
Winands, E., Adan, I., & Van Houtum, G. (2011). The stochastic economic lot scheduling problem: A survey. *European Journal of Operational Research*, *210*(1), 1–9.

Zapfel, G. (2010). *Metaheuristic search concepts*. New York: Springer.

Zhu, X., & Wilhelm, W. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE transactions*, *38*(11), 987–1007.

Zipkin, P. (1991). Computing optimal lot sizes in the economic lot scheduling problem. *Operations Research*, 56–63.

Curriculum Vitae



Personal Details

Name: Manuel Riel
E-mail: m@nuelriel.com
Nationality: Austrian
Date of birth: July 29, 1985

Education

2009-2011	University of Vienna Master of International Business Studies
since 2009	Technical University of Vienna B.A. of Business Information Technology
2008-2009	Shanghai University of Finance and Economics International Exchange Program
Aug-Sept 2007	Eurasia-Pacific Uninet Summer School Program "Modern China Studies for Scientists and Economists" in Shanghai and Beijing
2006-2009	University of Applied Sciences in Kufstein, Austria B.A. in International Business and Management
2000-2005	Commercial Highschool in Perg, Austria International Business and Culture Section

Work Experience

Since Oct 2009	Business Start-up: Facultas Consulting, GbR (in progress) Consulting in technical and business matters, done by students. Responsibilities include: • Customer Acquisition • Project Management
Since Oct 2008	Business Start-up: Hemd nach Mass.com, UG Distribution of tailor-made shirts over the internet. Responsibilities included: • Design and management of supply chain processes • Procurement and Relations Management with suppliers in Shanghai, China • Web development • Customer Service and Sales
Mar-Aug 2008	Proventis and Pacioli Institute for Financial Planning in Munich, Germany Corporate Finance Trainee. Responsibilities and tasks included: • Preparation and revision of financial planning models • Assistance with the development of presentations and investment documents • Development of detailed and complex sales planning models • Assistance with IT organization
Summer 2005	Galloway Sailing Center in Scotland, UK Climbing and Outdoor Activity Instructor. Responsibilities included: • Supervision and coordination of activities related to the climbing wall • Running other on- and off shore outdoor activities with different age groups

Language Skills

German (Native)
English (Fluent)
Chinese (Intermediate)
French (Elemantary)

Achievements and Interests

Since 2004	Climbing instructor with the Austrian Alpine Association
July 2006	Advanced Open Water Diver
Since 2006	Amateur Photographer
2005-2006	Military Service (4th Tank Support Division in Linz-Ebelsberg)
1993-2004	Boy Scout

Vienna, 5 Nov, 2011