1 Problembeschreibung

Bei synchronen Flow-Shop-Problemen handelt es sich um Produktionsplanungsprobleme, bei denen die zu produzierenden Güter (Jobs) auf einer kreisförmigen rotierenden Produktionsanlage produziert werden. Die Produktionsanlage besteht aus m Stationen, die sich mit der Anlage drehen. Außen, um die Anlage herum, befinden sich m fortlaufend nummerierte fixierte Maschinen $M_1, \ldots, M_m$, die die einzelnen Produktionsschritte durchführen. Dabei handelt es sich bei Maschine M_1 um das Einlegen des Jobs in die Anlage und bei Maschine M_m um die Entnahme des fertigen Produkts. Durch Rotation der Anlage werden die Stationen mit den auf ihnen befindlichen Jobs zur jeweils nächsten Maschine transportiert. Eine Rotation darf immer nur dann stattfinden, wenn alle Maschinen ihren Produktionsschritt an ihrem aktuellen Job durchgeführt haben. Auf diese Weise können die Jobs nur *synchron* zur nachfolgenden Maschine gelangen. Die Zeit, die zwischen zwei Rotationen vergeht, wird als *Zykluszeit* bezeichnet.

Die zu produzierenden Jobs sind gegeben durch die Menge $J = \{j_1, \ldots, j_n\}$ und die Prozesszeiten von Job j auf Maschine i sind durch p_{ij} gegeben. Ziel ist es, eine Reihenfolge $\pi \in J^n$ der Jobs zu erstellen, der die gesamte Produktionsdauer minimiert. Die Zykluszeiten c_k mit $1 \leq k \leq n+m-1$ berechnen sich wie folgt:

$$c_k = \max_{i=\max\{1,k-n+1\}}^{\min\{k,m\}} p_{i\pi_{k-i+1}}$$

Zusätzlich benötigen die Jobs Ressourcen aus einer Menge R, um in die Stationen eingelegt werden zu können. Diese Ressourcen können erst nach Fertigstellung eines Jobs, also nachdem er an Maschine M_m aus der Anlage genommen wurde, wiederverwendet werden. Sie sind allerdings nur in begrenzter Zahl vorhanden und im Allgemeinen ist nicht jede Ressource für jeden Job geeignet. Für $j \in J$ sei $\rho_j \subseteq R$ die Menge der Ressourcen, die für j geeignet ist. Umgekehrt sei für $r \in R$ mit $\iota_r \subseteq J$ die Menge der Jobs bezeichnet, für die r geeignet ist. An Maschine M_1 kann es daher notwendig sein, vor dem Einlegen des nächsten Jobs die Ressource zu wechseln, wenn auf der entsprechenden Station zuvor Job $j \in J$ mit Ressource $r \in \rho_j$ fertiggestellt wurde und nun Job $j' \in J$ eingelegt werden soll, wobei $r \notin \rho_{j'}$ ist.

Für die Ressourcen können folgende Situationen auftreten:

- Alle Ressourcen sind für alle Jobs geeignet, also $\rho_j = R$ für alle $j \in J$.
- Die Jobs lassen sich in disjunkte Gruppen unterteilen, so dass für alle Jobs aus einer Gruppe dieselbe Ressourcenmenge geeignet ist. Wenn also $\rho_i \cap \rho_j \neq \emptyset$, dann folgt $\rho_i = \rho_j$.
- Die Ressourcenmengen bilden Hierarchien. D.h., wenn $\iota_q \cap \iota_r \neq \emptyset$, dann folgt $\iota_q \subseteq \iota_r$ oder $\iota_r \subseteq \iota_q$.
- Die ρ_j sind beliebige Teilmengen von R.

Neben dem Wechsel von Ressourcen, der zusätzliche Zeit in Anspruch nimmt, können auch andere Formen von *Rüstkosten* auftreten. Z.B. kann es sein, dass

an einer Station zunächst einige Umstellungen vorgenommen werden müssen, bevor der neue Job eingelegt werden kann. Die Jobs können in Familien $\mathcal{F}$ eingeteilt werden, so dass beim Übergang zwischen zwei Jobs aus den Familien f und g die Rüstkosten s_{fg} auftreten. Diese Rüstkosten können

- sowohl vom Vorgänger als auch vom Nachfolger abhängig sein (s_{fg}),
- nur vom Nachfolger abhängig sein ($s_{fg} = s_g$) oder
- konstant sein ($s_{fg} = s > 0$).

Dabei wird jeweils für $s_{ff} = 0$ angenommen, dass also keine Rüstkosten innerhalb einer Familie auftreten.

Insgesamt gilt es also, neben dem Schedule π auch ein Mapping $f : J \to R$ zu finden, das jedem Job $j \in J$ eine Ressource $r \in \rho_j$ zuweist und folgenden Ansprüchen genügt:

- f muss zulässig sein in dem Sinne, dass beim Einlegen jedes Jobs $j \in J$ eine Ressource $r \in \rho_j$ verfügbar ist (d.h., dass r sich nicht gerade an anderer Stelle in der Anlage befindet).
- f und π zusammen sollen optimal sein in dem Sinne, dass die Summe aus den durch π und f definierten Zykluszeiten und Rüstkosten minimal ist.

2 Motivation

In [**?**] wurde gezeigt, dass dieses Problem schon $\mathcal{NP}$-schwer ist, wenn alle Ressourcen für alle Jobs geeignet sind und keine Rüstkosten auftreten. Versuche, dieses Problem – oder auch einige Spezialfälle davon – mit linearer Programmierung zu lösen, waren nur für sehr kleine Instanzen mit $n < 30$ erfolgreich, was weit hinter praktischen Anforderungen zurückliegt. Aufgrund der Komlexität des Problems sollen in dieser Arbeit zwei Dekompositionsansätze verfolgt werden:

1. Zunächst wird eine Reihenfolge π aufgestellt, ohne Ressourcen und Rüstkosten zu betrachten, und anschließend wird das Mapping f basierend auf π erstellt, ohne π noch zu verändern.
2. Es wird zuerst das Mapping f erstellt, so dass die Ressourcen zulässig zugewiesen sind und die Rüstkosten minimal sind. Anschließend werden, ohne f zu verändern, die Jobs so angeordnet, dass die Zykluszeiten möglichst minimal sind, und so π erstellt.

Beide Ansätze liefern natürlich im Allgemeinen keine optimalen Lösungen. Außerdem sind selbst die aus den Ansätzen resultierenden Teilprobleme teilweise noch $\mathcal{NP}$-schwer. Beispielsweise ist bei Ansatz (1) das Berechnen einer optimalen Reihenfolge π (soweit bekannt) nur in dem Spezialfall $m = 2$ mit dem Algorithmus von Gilmore und Gomory [**?**] in Polynomialzeit lösbar. Bei der anschließenden Zuweisung von Ressourcen ist noch unbekannt, ob ein polynomieller Algorithmus existiert. Dies herauszufinden ist eines der Ziele dieser Arbeit.

Da die Rüstkosten s_{fg} in dem gegebenen Praxisfall deutlich größer sind als die Prozesszeiten p_{ij} der Jobs auf den Maschinen, ist davon auszugehen, dass Ansatz (2) für diesen Praxisfall die besseren Lösungen erzielen wird. Nichtsdestotrotz ist auch der erste Ansatz interessant, da er in anderen Praxisbeispielen von Bedeutung sein kann.