**Informaticup 2023**

# Profit

von

**Lisa Binkert, Leopold Gaupe, Yasin Koschinksi**

RWU Hochschule Ravensburg-Weingarten University of Applied Sciences

Abgabedatum: 15.01.2023

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# 1 Einleitung

Der InformatiCup ist ein Wettbewerb, der von der Gesellschaft für Informatik jährlich veranstaltet wird. Studierende aller Fachrichtungen an Universitäten und Hochschulen in Deutschland, Österreich und der Schweiz dürfen daran teilnehmen.

Dieses Dokument beschreibt die Lösung für den InformatiCup 2023 des Teams “Die Schmetterlinge”. Das Team besteht aus Lisa Binkert, Leopold Gaube und Yasin Koschinski, welche alle an der RWU Hochschule Ravensburg-Weingarten University of Applied Science im Studiengang Master Informatik mit dem Profil Künstliche Intelligenz und Autonome Roboter eingeschrieben sind.

Im Kapitel 2 wird die Aufgabenstellung des Informaticup 2023 genauer erläutert. In Kapitel 3 werden verschiedene Lösungsansätze beschrieben, von denen nur ein Teil in die endgültige Lösung eingeflossen ist (Kapitel 4). Die Anwendung der Lösung wird in Kapitel 5, dem Benutzerhandbuch, beschrieben. In Kapitel 6 wird die Lösung evaluiert und bewertet, worauf in Kapitel 7 ein Fazit folgt.

# 2 Aufgabenbeschreibung

Die Aufgabe des InformatiCups 2023 ist das Lösen und Optimieren des rundenbasierten Spiels “Profit!”.

Das Spiel simuliert rundenbasierte Prozesse, in denen durch das Platzieren von verschiedenen Gebäude Ressourcen abgebaut und Produkte erstellt werden können. Das Herstellen von Produkten wird mit Punkten belohnt, wobei das Ziel das Maximieren dieser Punkte ist.

In den folgenden Abschnitten werden die Regeln und der Ablauf des Spiels sowie die Codierung des Spiels im JSON-Format kurz erläutert.

## 2.1 Spielregeln

Das Spielfeld besteht aus einem maximal 100x100 großen Rastergitter. Ein Raster ist entweder leer oder durch ein Objekt besetzt.

Die Abbildung reffig:task1 zeigt ein mögliches Spielfeld. In diesem Beispiel ist das Feld 30x20 groß und enthält drei Lagerstätten mit den Ressourcen Subtyp 0, 1 und 2, sowie zwei Hindernisse.

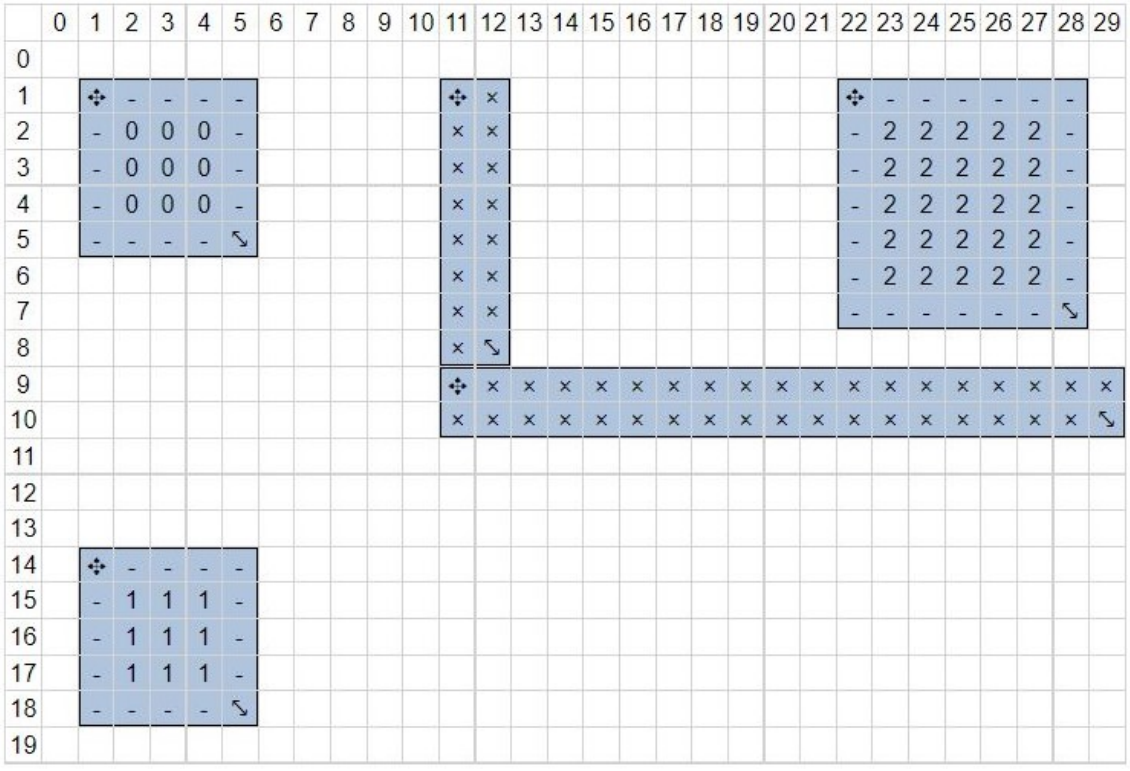


Abbildung 1: Beispielumgebung eines Profit-Spiels

Das linke obere Raster befindet sich an der Stelle (0,0), die rechte untere (Breite-1, Höhe-1). Die Größe des Spielfeldes ist pro Spiel vorgegeben.

Zu Beginn des Spiels sind bereits Hindernisse und Lagerstätten mit Ressourcen vorhanden. Ebenso werden die maximale Rundenanzahl und die Produkte, die produziert werden können, festgelegt.

Die Menge an Ressourcen einer Lagerstätte wird durch die Größe der Lagerstätte festgelegt. Diese wird durch die Menge an Rastern x 5 festgelegt. Ist also eine Lagerstätte $3 \cdot 3$ dann enthält es $3 \cdot 3 \cdot 5 = 45$ Ressourcen eines bestimmten Subtyps. Insgesamt gibt es 8 Subtypen (0-7).

Hindernisse im Spielfeld stellen Felder dar, in denen kein anderes Gebäude gebaut werden kann. Diese sind beliebig groß, haben aber immer eine rechteckige Form.

**Produkt**
Für jedes Spiel ist mindestens ein Produkt definiert, maximal acht. Ein Produkt benötigt eine beliebige Kombination der acht Ressourcen. Die benötigte Menge der jeweiligen Ressource ist ebenfalls definiert. Beispielsweise kann zur Herstellung von Produkt 0 dreimal die Ressource 0 und einmal die Ressource 1 benötigt werden. Jedes Produkt gibt eine bestimmte Anzahl an Punkte.

**Mine**
Um die Ressourcen in den Lagerstätten abzubauen, gibt es Minen. Sie ist 4x2 oder 2x4 Felder groß und hat vier Subtypen, die die Rotation der Mine bestimmen. Die Abbildung 2 zeigt die vier verschiedenen Subtypen der Mine. Jede Mine hat

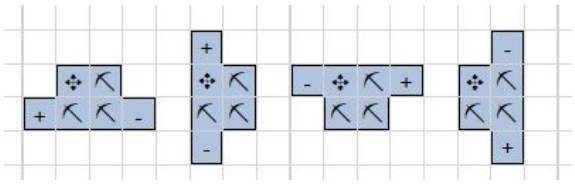

Abbildung 2: Darstellung der vier Minen-Subtypen

einen Eingang (+) und einen Ausgang (-). Der Eingang muss an einem Ausgang einer Lagerstätte anliegen, um die Ressourcen abzubauen. An dem Ausgang können Eingänge von Förderbändern, Verbindern oder Fabriken anliegen.

**Förderband**
Um Ressourcen von einer Lagerstätte zu einer Fabrik zu befördern, können Förderbänder genutzt werden. Förderbänder sind nicht zwangsläufig notwendig. Sie stellen zusätzliche Verbindungsstücke zwischen Mine und Verbinder, Mine und Fabrik oder Verbinder und Fabrik dar.

Diese haben wie andere Objekte auch einen Eingang (+) und einen Ausgang (-). Ein Förderband ist entweder 3 oder 4 Raster lang und kann, wie in der Abbildung 3 gezeigt, je in vier Subtypen mit unterschiedlicher Ausrichtung verwendet werden. Somit hat das Förderband insgesamt acht Subtypen. Im Gegensatz zu allen anderen

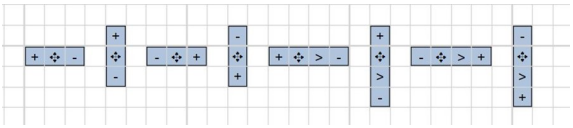

Abbildung 3: Darstellung der acht Förderband-Subtypen

Objekten dürfen sich Förderbänder auch kreuzen.

**Verbinder**
Wenn ein Produkt mehrere Ressourcen benötigt, können diese mit einem Verbinder zusammengeführt und gemeinsam zur Fabrik befördert werden.
Ein Verbinder hat drei Eingänge (+) und einen Ausgang (-). Auch hier gibt es vier Subtypen, die jeweils die Rotation des Verbinders bestimmen (Abbildung 4).

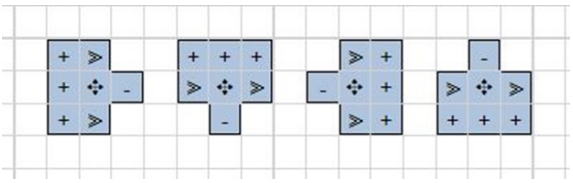

Abbildung 4: Darstellung der acht Förderband-Subtypen

**Fabrik**
Für jede Fabrik ist definiert, wie viele und welche Ressourcen gebraucht werden, um ein Produkt herzustellen. Da es maximal acht Produkte geben kann, gibt es von der Fabrik insgesamt acht verschiedene Subtypen, für jedes Produkt einen. Jede Fabrik ist 5x5 groß. Die äußeren Raster sind Eingänge für Ressourcen, insgesamt 16.
Mit einer Kombination aus Mine, Förderband und Verbinder werden die Ressourcen von den Lagerstätten zur Fabrik befördert. Die Abbildung 5 zeigt, wie so ein Aufbau aussehen kann. Im Beispiel benötigt das Produkt 0 die Ressourcen 0 und 1.

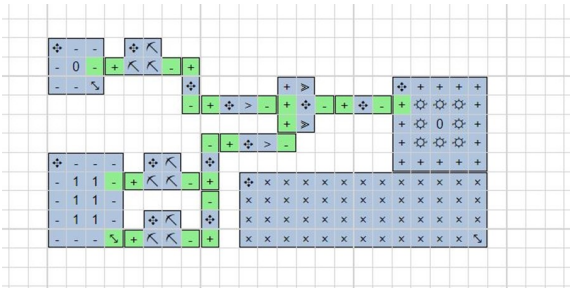

Abbildung 5: Darstellung der acht Förderband-Subtypen

## 2.2 Spielablauf

Das Spiel läuft rundenbasiert ab. Jede Runde beginnt mit einer “Beginn der Runde”-Aktion und endet mit einer “Ende der Runde”-Aktion. Die folgende Tabelle definiert

| Objekt | Beginn der Runde | Ende der Runde |
|---|---|---|
| Mine | Nimmt Ressourcen auf | Gibt angenommene Ressourcen weiter |
| Förderband | Nimmt Ressourcen auf | Gibt angenommene Ressourcen weiter |
| Verbinder | Nimmt Ressourcen auf | Gibt angenommene Ressourcen weiter |
| Fabrik | Nimmt Ressourcen auf | Produziert so viele Produkte wie Ressourcen da sind |
| Lagerstätte | - | Gibt bis zu 3 Ressourcen an jeden Eingang einer Mine |

Tabelle 1: Beschreibung der Aktionen "Beginn der Runde"und "Ende der Runde"

die Aktionen der einzelnen Objekte. Das Spielfeld hat neben der Feldgröße und den vorhandenen Objekten auch das Attribut "turns". Dieses Attribut gibt die maximal erlaubte Anzahl an Spielrunden an.

Pro Runde werden Ressourcen von Lagerstätten abgebaut und Stück für Stück über die gebauten Förderbänder und gegebenenfalls Verbinder zu einer Fabrik befördert. Wenn eine oder alle Ressourcen eines Produktes abgebaut wurden und damit kein weiteres Produkt mehr produziert werden kann, stoppt die Produktion dieser Fabrik. Ist das bei allen gebauten Fabriken der Fall, so endet das Spiel und der Spieler erhält eine Gesamtpunktzahl und die Anzahl der dafür benötigten Runden.

Das Ziel des Spiels ist es, die Punktezahl zu maximieren und die Rundenanzahl zu minimieren.

## 2.3 Darstellung in JSON

Der Input des Spiels erfolgt im JSON-Format, in dem Lagerstätten, Hindernisse und Produkte definiert sind. Das JSON für das Beispiel in Abbildung 6 sieht wie folgt aus: Das Spielfeld wird durch die angegebene Breite (width) und Höhe(height) de-


```
{"width":30,"height":20,"objects":[
{"type":"deposit","x":1,"y":1,"subtype":0,"width":5,"height":5},
{"type":"deposit","x":1,"y":14,"subtype":1,"width":5,"height":5},
{"type":"deposit","x":22,"y":1,"subtype":2,"width":7,"height":7},
{"type":"obstacle","x":11,"y":9,"width":19,"height":2},
{"type":"obstacle","x":11,"y":1,"width":2,"height":8}],
"products":[{"type":"product","subtype":0,"resources":[3,3,3,0,0,0,0,0]
,"points":10}],"turns":50}
```


Abbildung 6: JSON Darstellung einer Aufgabe

finiert. Es enthält fünf verschiedene Objekte (objects), drei Lagerstätten und zwei Hindernisse. Jedes dieser Objekte besitzt eine x- und y-Koordinate, die die Position im Feld bestimmt, sowie eine Höhe und eine Breite. Lagerstätten haben einen Subtyp, welcher die hier verfügbare Ressource festlegt. Nach den Objekten werden die Produkte definiert. Im JSON sind hierfür der Subtyp, die benötigten Ressourcen

| **Objekt** | **JSON** |
|---|---|
| Mine | {”type”:”mine”, ßubtype”:0, ”x”:0, ”y”:0} |
| Förderband | {”type”:”conveyor”, ßubtype”:0, ”x”:0, ”y”:0} |
| Verbinder | {”type”:”combiner”, ßubtype”:0, ”x”:0, ”y”:0} |
| Fabrik | {”type”:”factory”, ßubtype”:0, ”x”:0, ”y”:0} |
| Lagerstätte | {”type”:”deposit”, ßubtype”:0, ”x”:0, ”y”:0, ”width”:1, ”height”:1} |
| Hindernisse | {”type”:öbstacle”, ”x”:0, ”y”:0, ”width”:1, ”height”:1} |

Tabelle 2: Beschreibung der Aktionen ”Beginn der Runde”und ”Ende der Runde”

(ressources) pro Produkt und die Anzahl an Punkten (points) pro Produkt angegeben. Die Position der Ressource im Array bestimmt den jeweiligen Subtyp.

Jedes Objekt kann durch einen JSON-String dargestellt werden, der die Eigenschaften des Objekts definiert. In folgender Tabelle wird für jeden Typ ein beispielhafter JSON-String aufgeführt:

# 3 Lösungsansätze

## 3.1 Reinforcement Learning

## 3.2 Deep-Q-Learning

## 3.3 Actor-Critic

## 3.4 Mone-Carlo-Search-Tree

## 3.5 Regelbasierter Ansatz

# 4 Lösungsumsetzung

## 4.1 Programmiersprache und Bibliotheken

## 4.2 Grundidee

## 4.3 “Profit!” Umgebung

### 4.3.1 Buildings

### 4.3.2 Spiel-Simulation

### 4.3.3 Optimale Score

### 4.3.4 Testing

## 4.4 Untergeordneter Agent

## 4.5 Übergeordnete Agent

## 4.6 Wartbarkeit

# 5 Benutzerhandbuch

# 6 Evaluation

# 7 Fazit

# Literaturverzeichnis