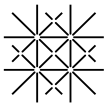

University of Basel

Optimierungsanfragen für Town Building Spiele mithilfe von LP- & IP-Solver

Bachelorarbeit

Philosophisch-Naturwissenschaftliche Fakultät
Departement Mathematik und Informatik
Artificial Intelligence
https://ai.dmi.unibas.ch/

Beurteiler: Prof. Dr. Malte Helmert
Zweitbeurteilerin: Dr. Salomé Eriksson

Mateusz Palasz
mateusz.palasz@stud.unibas.ch
2017-058-777

16.08.2021

Anerkennung

Allem voran möchte ich mich bei Dr. Salomé Eriksson für ihre Zeit und Lenkung während den letzten drei Monaten bedanken. Die zahlreichen Hinweise, sowie Ratschläge waren wegweisend für meine Arbeit. Zudem bin ich Prof. Dr. Malte Helmert sehr dankbar mir die Möglichkeit gegeben zu haben an einem solchen interessantem Thema, für die Fachgruppe AI der Universität Basel, arbeiten dürfen.
Abschließend bin ich meinen Eltern und Freunden für die großartige Unterstützung sehr verbunden. Sie waren nicht nur stets an meiner Seite, sondern haben mich Tag für Tag motiviert meine Ziele zu verfolgen und an mich zu glauben.

Abstrakt

Lineare Programmierung ist eine mathematische Modellierungstechnik, bei der eine lineare Funktion, unter der Berücksichtigung verschiedenen Beschränkungen, maximiert oder minimiert werden soll. Diese Technik ist besonders nützlich, falls Entscheidungen für Optimierungsprobleme getroffen werden sollen. Ziel dieser Arbeit war es ein Tool für das Spiel Factory Town zu entwickeln, mithilfe man Optimierungsanfragen bearbeiten kann. Dabei ist es möglich wahlweise zwischen diversen Fragestellungen zu wählen und anhand von LP-\ IP-Solvern diese zu beantworten. Zudem wurden die mathematischen Formulierungen, sowie die Unterschiede beider Methoden angegangen. Schlussendlich unterstrichen die generierten Resultate, dass LP Lösungen mindestens genauso gut oder sogar besser seien als die Lösungen eines IP.

Inhaltsverzeichnis

Anerkennung **ii**

Abstrakt **iii**

1 Einleitung **1**

2 Hintergrund **2**
2.1 Factory Town 2
2.2 Lineare Optimierung 4
2.2.1 Bestandteile 4
2.2.2 Geschichtlicher Hintergrund 5
2.3 Integer Optimierung 5

3 Linear Programming **6**
3.1 Mathematische Formulierung 6
3.2 Beispiel 7

4 Implementierung **9**
4.1 Google OR-Tool 9
4.1.1 Glop 9
4.1.2 SCIP 9
4.1.3 Vorgehensweise 10
4.2 Aufbau Datenbank 10
4.3 LP & IP 11
4.3.1 Arbeitsverteilung für Items 12
4.3.2 Ranking vom Verkaufswert 14
4.3.3 Höchstmöglicher Profit einer Münzenart 14
4.3.4 Gewichtung der Münzenarten 16

5 Gegenüberstellung der Resultate LP & IP **19**

6 Fazit **21**

Literaturverzeichnis **22**

Declaration on Scientific Integrity **24**

1 Einleitung

Ein altbekanntes lineare Problem ist das „Diät-Problem“ [17] entwickelt von George J. Stigler im Jahr 1944, dass bei der Suche nach einer kostengünstigen Ernährung, die den Nährstoffbedarf eines US-Armee-Soldaten decken soll, helfen sollte. Die Stigler-Diät wurde wegen ihres Mangels an Abwechslung und Schmackhaftigkeit viel belächelt, seine Methodik wurde jedoch gelobt und gilt als eine der frühesten Arbeiten in der linearen Programmierung. Gekennzeichnet wurde ihre Leistung durch eine lange Geschichte, während die meisten vergleichbaren Lösungen für Ernährungsprobleme erst im Jahr 2000 oder später entwickelt wurden, als Computer mit großen Rechenkapazitäten weit verbreitet waren und Werkzeuge zur linearen Programmierung (LP) bereits entwickelt wurden.
LP sind auf eine Vielzahl von Ernährungsproblemen anwendbar, von Nahrungsmittelhilfe, nationalen Ernährungsprogrammen und Ernährungsleitlinien bis hin zu individuellen Themen. So auch an die Fragestellungen, wie nach der perfekten Arbeitsverteilung für Items, oder den höchstmöglichen Profit einer Münzart. Diese Fragestellungen werden im Kontext des Spiels Factory Town in dieser Arbeit angegangen. Lineare Programmierung ist ein wichtiges Werkzeug für kombinatorische Suchprobleme, nicht nur, weil sie eine große Klasse wichtiger Probleme effizient löst, sondern auch, weil sie der grundlegende Block einiger elementarer Techniken in diesem Bereich ist.
Bei der ganzzahligen Programmierung werden lineare Programme um die zusätzliche Forderung ergänzt, dass einige oder alle Variablen nur ganzzahlige Werte annehmen dürfen. Durch diese gleichermaßen wichtige wie einfach erscheinende Forderung wird die numerische Lösbarkeit, zugunsten der Vielfältigkeit an realen Probleme, erheblich erschwert.
Diese Arbeit soll sowohl das Grundkonzept, als auch Vor-\ Nachteile beider Methoden, anhand von den Resultaten, hervorgehend aus Anwendungsbeispielen eines entwickelten Tools, nahelegen.

2
Hintergrund

Dieses Kapitel soll eine Basis, für die darauf folgenden Themen legen. Insbesondere werde ich auf das Spiel Factory Town [9] eingehen, sowie auf das Grundkonzept von linear programming (LP) \ integer programming (IP).

2.1 Factory Town

Ausgehend von einem kleinen Town Center ist das Ziel des Spiels, das Dorf mittels Automatisierung zu einer riesigen Fabrik zu entwickeln. Dabei fängt man mit wenigen Arbeitern an und sammelt die ersten Ressourcen. Um schließlich mit mehr Arbeitern noch schneller Rohstoffe abzubauen, müssen weitere Häuser errichten werden. Durch das Aufbauen von Gebäuden, werden die ersten Items automatisiert. Dafür werden Förderbänder für Materialien errichtet, sowie neue Technologien erforscht.

Gebäude sind ein Kernelement von Factory Town. Grundsätzlich gibt es zwei Arten von Gebäuden, einerseits Markt Gebäude, die für den Verkauf von Gütern zuständig sind, anderseits Gebäude, die zur Herstellung von Items dienen. Durch die Vielfältigkeit entsteht eine Vielzahl von Möglichkeiten, ein Dorf auf eigene Art und Weise zu verbessern. Gelegentlich ermöglicht der Bau eines neuen Gebäudes das Errichten anderer. Beim Platzieren der meisten Gebäude werden Arbeiter benötigt, die die Produktionsgeschwindigkeit erhöhen können, indem mehrere hinzugefügt werden.

Arbeiter sind flexibel einsetzbar. Sie können entweder für den Transport genutzt werden, oder zur Steigerung der Produktionsgeschwindigkeit. Zumal das Spiel auf optimierter Automatisierung beruht, sind Arbeiter ab einem gewissen Punkt nicht mehr für den Transport zuständig. Dementsprechend fungieren sie als Grundvariable, welche die Produktionsgeschwindigkeit beeinflusst. Arbeiter können Aufgaben nicht parallel erledigen, wodurch eine klare Zuweisung an eine Auftrag entsteht. Infolgedessen werden Arbeiter fix an ein bestimmtes Gebäude gebunden.
Zum Vereinfachung wurde in meinem Tool keine Unterteilung zwischen Gebäuden und Arbeitern gemacht. Daraus folgt ein Arbeiter, der an einen Aufragt gebunden ist, ist gleich-

gestellt mit dem für den Auftrag verantwortlichen Gebäude. Sollte es sich herausstellen, dass z.B. zwei Arbeiter für den Auftrag notwendig sind, so werden zwei Gebäude benötigt. Häuser können die Kapazität von Arbeitern erhöhen.

Förderbänder jeglicher Art sind optimal, da sie einerseits die Ressourcen schneller voranbringen, anderseits ermöglichen sie, dass Arbeiter einzig als Faktor für die Produktionsgeschwindigkeit gesehen werden können.
Abhängig vom variierenden Aufbau der Welt, sowie der Benutzerspezifischen Art der Anordnung von Gebäude und Förderbänder, wurde die Transportzeit, beziehungsweise der „Size Faktor“ im Tool vernachlässigt.

Items können in unterschiedlichen Kategorien eingeordnet werden. Darunter fallen auch die natürliche Ressourcen, welche als Atome gelten und eine Basis für alle anderen Items bilden. Grundsätzlich unterscheiden sich die Items in ihren Eigenschaften, dazugehören die Herstellungszeit, der Verkaufswert, die Verbrauchszeit, die zuvor erwähnte Kategorie, wie auch die Zutaten[1] für die Herstellung, welche ein Rezept bilden. Die genau Bedeutungen der einzelnen Eigenschaften, werden im Abschnitt 4.2 erklärt.

Häuser werden benötigt, um die Bevölkerung aufzubauen und Münzen zu sammeln. Sollte die Kapazität von Arbeitern erreicht worden sein, so ist es mögliche diese zu updaten[2], um zusätzliche Arbeiter zu erzeugen. Ansonsten können weitere Häuser erzeugt werden, wodurch sich zusätzlich der Konsum von Gütern erhöht, was im Umkehrschluss positive Auswirkungen auf den Entstehungsprozess von Münzen hat.

Münzen existieren in vier verschiedenen Arten: Gelb, Rot, Blau und Lila. Sie werden benötigt um Fabriken und Fördersysteme zu bauen, als auch um Gebäude aufzuwerten. Zudem sind sie essenziell um Forschung zu betreiben. Verdienen kann man Münzen, in dem Items an Märkte und Häuser verkauft werden, die ihren Bedarf abdecken.

Glück & Konsum hängen davon ab, wie viele unterschiedliche Bedürfnisse eines Hauses befriedigt werden. Zusammenaddiert gibt es 51 Arten von Bedürfnissen, die in fünf Kategorien unterteilt sind. Wird ein Item an ein Haus verkauft, so müssen die Bewohner, diesen in der vorgegeben Verbrauchszeit konsumieren. Es können nicht mehr gleichartige Items konsumiert werden, als die Anzahl der Häuser, geteilt durch die Verbrauchszeit. Umso mehr verschiedene Kategorie abgedeckt werden können, desto mehr steigert sich das Glück der Bewohner. Dies spiegelt sich positiv auf die Produktionsgeschwindigkeit wider. Jener Glücks-Faktor wurde im Tool außer acht gelassen.

Forschen an der Schule, ermöglicht das neue Arten von Rezepten und Gebäuden freigeschaltet werden können.

[1] Zutaten verkörpern die für die Herstellung benötigten Items.
[2] Die resultierende Anzahl an Arbeiter setzt sich aus dem (Level der Häuser · 2) · Anzahl der Häuser + 4 zusammen.

2.2 Lineare Optimierung

Die lineare Optimierung ist eines der hauptsächlich genutzten Verfahren im Operations Research[3] und befasst sich mit der Optimierung von linearen Zielfunktionen, unter der Einschränkung von linearen Gleichungen und Ungleichungen. Diese Art von Problemen treten in vielen Natur- und Ingenieurwissenschaftlichen Bereichen auf und dienen zur Unterstützung ökonomischer Entscheidungsprozesse. Grundsätzlich können sie überall angewandt werden, wo lineare Zielfunktion, unter Einhaltung von linearen Restriktionen, maximiert \ minimiert werden sollen [18]. Die lineare Optimierung kann sowohl mathematisch als auch graphisch gelöst werden.

2.2.1 Bestandteile

Klassische LP & IP haben prinzipiell einen einheitlichen Aufbau, der aus folgenden Komponenten besteht.

Definition 1. *Die* ***Variablen***[4] *beschreiben eine kompakte Darstellung der Menge aller möglichen Werte der Lösungen [5].*

Definition 2. *Eine* ***Nebenbedingungen*** *(Restriktionen) ist eine Ungleichung, die definiert, wie die Werte der Variablen in einem Problem begrenzt werden.*

Definition 3. *Der Bereich, der durch das System von Restriktionen begrenzt ist, wird als* ***zulässiger Bereich*** *bezeichnet. Es stellt die möglichen Werte der Variablen dar, die alle Einschränkungen erfüllen.*

Definition 4. *Die* ***Bedingung der Nichtnegativität****, grenzt zusätzlich den zulässigen Bereich ein. Alle Variablen innerhalb der linearen Optimierung müssen gleich oder größer als Null sein.*

Ergänzend zur Definition 4 zu erwähnen wäre, dass LP\ IP's vorkommen die auch negative Werte zulassen, allerdings werden diese in diesem Zusammenhang nicht wahrgenommen. Das Auffinden des zulässigen Bereichs reicht nur aus, um die möglichen Lösungen eines Problems anzugeben. Das Ziel der linearen Programmierung besteht darin, die beste Lösung für ein Problem zu finden. Existiert keine optimale Lösung, so ist auch möglich, dass das lineare Programm nicht lösbar ist.

Definition 5. *Die* ***Zielfunktion*** *ist eine Funktion, die eine Größe definiert, die minimiert oder maximiert werden sollte. Die Argumente der Zielfunktion sind dieselben Variablen, die in den Einschränkungen verwendet werden.*

Der Begriff linear erklärt sich dadurch, dass die Zielfunktion und sämtliche Restriktionen linear sind.

[3] OR bezeichnet die Entwicklung und den Einsatz von mathematischen Verfahren zur Unterstützung von Entscheidungsprozessen.

[4] Werden gelegentlich Entscheidungsvariablen genannt, weil das Problem darin besteht, zu entscheiden, welchen Wert jede Variable annehmen soll.

2.2.2 Geschichtlicher Hintergrund

Der Beginn der linearen Optimierung ist auf das Ende der 1930er Jahren zurückzuführen, als der sowjetischen Mathematischer Leonid Kantorowitsch seinem Buch „Mathematische Methoden in der Organisation und Planung der Produktion“ [13] veröffentlichte.
Erst als George Dantzig erkannte, dass sich viele praktische Beschränkungen durch lineare Ungleichungen beschreiben ließen, setzte er eine lineare Zielfunktion ein. Insbesondere etablierte er damit eine klare Trennung zwischen dem Ziel der Optimierung und den Mitteln zur Lösung des Planungsproblems [6].
Das älteste und bekannteste Lösungsverfahren für LP ist das, von George B. Dantzig im Jahr 1947 entwickelte, Simplex-Verfahren [11], welches heute immer noch eines der meistgenutzten Verfahren zur Lösung linearer Programme ist [16].

2.3 Integer Optimierung

Übereinstimmend zu der linearen Optimierung beschäftigt sich die integer Optimierung mit der Optimierung linearer Zielfunktionen über einer Menge, die durch lineare Gleichungen und Ungleichungen eingeschränkt ist. Mit einem Unterschied, alle Variablen müssen ganzzahlig sein. Sollte dennoch ein Teil der Variablen Gleitkommazahlen sein, so spricht man von Mixed Integer Programs (MIPs).

Definition 6. *Ein integer Programm ist ein lineares Programm, in dem alle Variablen ganze Zahlen sein müssen.*

Die Verwendung von integer Variablen erweitert den Umfang an Modellierungsmöglichkeiten für praktische Optimierungsprobleme, die definiert und gelöst werden können, erheblich. Nichtsdestoweniger auf Kosten von mehr Komplexität. Dessen ungeachtet können beispielsweise Alltagsgegenstände, oder die Mehrheit der üblichen Ressourcen nicht als reelle Werte dargestellt werden, was einen elementaren Vorteil mit sich bringt. Zusätzlich ermöglichen sogenannte Binärvariablen[5] die Entscheidungsfindung.
Lineare Programme lassen sich trotz sehr vielen Variablen und Nebenbedingungen effizient in polynomialer Zeit berechnen, zum Beispiel mithilfe von dem Karmarkar Algorithmus [14]. Leider ist es nicht mehr der Fall, wenn die Variablen ganzzahlige Werte annehmen müssen. Dies liegt daran, dass viele Kombinationen spezifischer ganzzahliger Werte für die Variablen getestet werden müssen und jede Kombination die Lösung eines „normalen“ linearen Optimierungsproblems erfordert. Die Anzahl der Kombinationen kann mit der Größe des Problems exponentiell ansteigen. Daraus resultiert, dass es sich hierbei um ein NP-schweres Problem handelt [3].
Zudem wäre zu beachten, dass ein gegebenes IP in ein LP relaxiert werden kann, indem man die Restriktion, dass alle Variablen ganzzahlig sind, aufhebt. Die optimale Lösung dieser Relaxierung ist dann mindestens genau so gut oder sogar besser als die Lösung des originalen IPs, da der zulässige Bereich in der Relaxierung größer ist [10].

[5] Ganzzahlige Variablen, die sich auf 0 oder 1 beschränken

3
Linear Programming

Der Kern dieses Kapitels ist die allgemeine Mathematische Formulierung von linearer Programmierung, sowie eine Ergänzung für integer Probleme. Abgerundet wird dies mit einem Beispiel, inklusive Gegenüberstellung der Resultate von LP und IP.

3.1 Mathematische Formulierung

Ein lineares Problem lässt sich grundsätzlich durch wenige, zuvor erwähnte Faktoren, verallgemeinern. Demnach sollten eine Matrix $A \in \mathbb{R}^{m,n}$, zusätzlich zu zwei weiteren Vektoren $b \in R^{m,1}$ und $c \in R^{1,n}$ gegeben sein. Diese lassen sich normalerweise aus den Anwendungsaufgaben entnehmen. Ergänzend fehlen nur noch die Variablen, dargestellt durch ein nicht mit negativen Einträgen versehenen Vektor $x \in R^n$, der zulässige Lösungen repräsentiert. Diese werden mittels Nebenbedingungen $Ax \leq b$, welche äquivalent zur gegeben Notation 3.1 sind, begrenzt. Zusammen definieren sie den zulässigen Bereich.

$$\begin{array}{cccc} a_{11}x_1 & +\ldots & +a_{1n}x_n & \leq b_1 \\ a_{21}x_1 & +\ldots & +a_{2n}x_n & \leq b_2 \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}x_1 & +\ldots & +a_{mn}x_n & \leq b_m \end{array} \tag{3.1}$$

Das Ziel besteht darin, in der Menge aller möglichen Werte des Vektors x, eine Kombination zu finden, die das Standardskalarprodukt

$$cx = c_1x_1 + \ldots + c_nx_n \tag{3.2}$$

entsprechend maximiert. Jede Lösung aus dieser Lösungsmenge, ist durch die Zielfunktion an eine Zahl cx[6] gebunden. Dementsprechend besagt die Zielfunktion, dass eine Lösung mit einem größeren Zielfunktionswert besser ist, als jede andere Lösung mit einem kleineren Zielfunktionswert. Eine solche Lösung nennt man optimal [15]. Sollten es sich um ein Minimierungsproblem handeln, so wird c mithilfe eines Koeffizienten -1 ergänzt. Diese Art von

[6] Der Zielfunktionswert wird, für eine bestimmte zulässige Variable Kombination, durch cx repräsentiert.

Optimierungsproblemen werden üblicherweise, in der Standardform[7] abgekürzt [6].

$$\max\{cx \mid Ax \leq b, x \geq 0\} \tag{3.3}$$

Nebenbedingungen können in der Regel in linearer ($Ax = b$), oder in gebundener Form ($l \leq x \leq u$) auftreten[7]. Die Variablen $l, u \in \mathbb{R}$ symbolisieren Konstanten, die für den „lower and upper bound“[8] stehen. Zudem ist es einfach eine Gleichheitsbedingungen in ein Paar von Ungleichheitsbedingungen umzuwandeln[12].

$$a_i x = b_i \Longleftrightarrow a_i x \leq b_i \text{ und } -a_i x \leq -b_i \tag{3.4}$$

Sollte es sich beim Optimierungsproblem konkreterweise um ein ganzzahliges Problem handeln, so hat dieses die gleiche Mathematische Formulierung wie zuvor spezifiziert. Allerdings mit einem Unterschied, dass die Variablen ganzzahlig sein müssen.

$$\max\{cx \mid Ax \leq b, x \geq 0, x \in \mathbb{Z}^n\} \tag{3.5}$$

Die Bedingung $Ax \leq b$ ist komponentenweise, für alle Zeilen i in der Matrix A, zu verstehen[8].

$$a_i \cdot x = \sum_{j=1}^{n} a_{ij} x_j \leq b_i \tag{3.6}$$

Erwiesenermaßen gilt für Optimierungsproblemen genau einer der drei Alternativen:

- Das Problem ist unbeschränkt, daher findet man für jede zulässige Lösung stets eine weitere zulässige Lösung mit einem besserem Zielfunktionswert.
- Das Problem ist unzulässig, insofern existiert überhaupt keine Lösung, die alle Restriktionen erfüllt.
- Es existiert eine optimale Lösung mit endlichem Zielfunktionswert.

[15]

3.2 Beispiel

Das durch die Abbildung 3.1 veranschaulichte Beispiel wird durch folgendes Optimierungsproblem

$$\begin{aligned} \max\ & x + y \\ 1.75 \cdot x + y &\leq 4.5 \\ x &\leq 2 \\ x, y &\geq 0 \\ x, y &\in \mathbb{R} \end{aligned}$$

[7] Auch Ungleichungsform bezeichnet.
[8] Untere \ obere Schranke

dargestellt. Die Variablen x, y wurden hierbei als reelle Zahlen deklariert und erzeugen, durch die Beachtung der Restriktionen und der Nichtnegativitätsbedingung, den grünen zulässigen Bereich. Als Kontrast dienen die blauen Punkte, die den wesentlich kleineren zulässigen Bereich erzeugen würden, falls die Variablen $x, y \in \mathbb{Z}$ wären. Ziel der Optimierung wäre es nun die Zielfunktion $x + y$ zu maximieren und auf Grund dessen den höchsten Zielfunktionswert ausfindig zumachen.

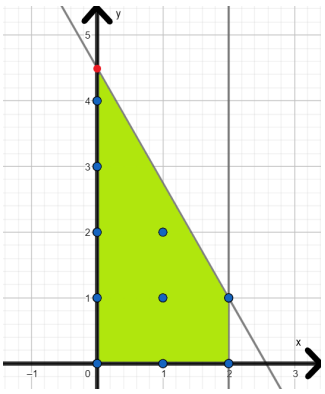

Abbildung 3.1: Ein einfaches Beispiel von einem Optimierungsproblemen.

Die Optimallösung für dieses lineare Problem, wir anhand vom Punkt $(0, 4.5)$ dargelegt, dieser wir in der Abbildung 3.1 rot dargestellt und hat einen Zielfunktionswert von $cx = 4.5$. Alternativ würden bei einem integer Problem, mit ganzzahligen Variablen, ausschließlich die blauen Punkte als Optimallösung in Frage kommen, was zu einer optimalen Variablenbelegung $IP_{opt} = (0, 4)$ und einem Zielfunktionswert von 4 führt, also einem tieferen Wert als beim entsprechenden LP.

4
Implementierung

In erster Linie wird sich alles in diesem Kapitel um die Implementierung drehen. Darunter fällt einerseits das von Google entwickelte OR-Tool, andererseits wird der Aufbau der Datenbank thematisiert. Abrundend werden die Formeln für das LP & IP, inklusive der dazugehörigen Ausgabe, behandelt.

4.1 Google OR-Tool

Das von Google entwickelte OR-Tool [4] ist eine Open-Source-Software darauf abgestimmt kombinatorische Optimierungsprobleme im Bereichen von Vehicle Routing, Flows, integer\ linear Programming, sowie constraint-Programmierung anzugehen. Insbesondere wenn die beste Lösung für ein Problem aus einer sehr großen Menge herausgefiltert werden soll.
Wobei es dem Benutzer frei zur Auswahl steht, ob er in C++, Python, C#, oder Java implementiert. Zudem ist es möglich aus einer Vielzahl von unterschiedlichen Solver zu selektieren.

4.1.1 Glop

Glop ist Google's hausinterner Solver, der hauptsächlich für lineare Programmierung verwendete wird. Entwickelt wurde er vom Google's Operations Research Team, welches es im Jahr 2014 als Teil von Google OR veröffentlichten. Laut Herstellerangaben ist er schnell, speichereffizient und numerisch stabil[9] [1]. Das Ziel von Glop ist es den optimalen Wert einer linearen Zielfunktion zu finden, wenn eine Reihe von linearen Ungleichungen als Einschränkungen gegeben sind.

4.1.2 SCIP

SCIP ist laut Herstellerangaben einer der schnellsten non-commercial Solver für Mixed Integer Programming (MIP) [2]. Die erste Version wurde im Jahr 2005 released, welche weiterhin

[9] In der Numerik steht ein stabiles Verfahren für Unempfindlichkeit gegenüber kleinen Störungen der Daten, samt von Rundungsfehlern.

bis zum heutigen Tag weitergeführt wird. Im Gegenteil zu Glop, was in C++ implementiert wurde, ist SCIP als aufrufbare C-Bibliothek umgesetzt worden und stellt C++ Wrapper-Klassen für Benutzer-Plugins bereit. Es kann auch als eigenständiges Programm verwendet werden, um MIP samt anderen nicht linearen Programmen zu lösen.
MIP Solver eignen sich bestens für Probleme, die als Standard-LP eingerichtet werden können, jedoch mit beliebigen ganzzahligen Variablen.

4.1.3 Vorgehensweise

Wenn es um die Vorgehensweise bei Optimierungsproblemen geht, haben diese grundsätzlich identischen Aufbau. Sie unterscheiden sich lediglich bei der Auswahl vom Solver und zugleich bei der Deklarierung der Variablen. Allerdings wird ausschließlich differenziert, ob die Variablen $x \in \mathbb{R}$, oder $x \in \mathbb{Z}$ sind. Die Anzahl der Variablen ist bei der selben Problemstellung, identisch.

1. Wählen Sie zwischen den zahlreichen unterschiedlichen Solvern, darunter fallen auch Glop und SCIP.
2. Erzeugen Sie die benötigten Variablen. Einerseits muss entscheiden werden, ob es sich um ganzzahlige(IntVar), oder reelle(NumVar) Variablen handelt. Anderseits, in welchen Wertebereich sie sich bewegen dürfen, üblicherweise wir ein Bereich $\mathbb{W} = [0, \infty)$ angegeben was von vornherein die Bedingung der Nichtnegativität erfüllt.
3. Definieren Sie als Nächstes die Einschränkungen für die Variablen.
4. Stellen Sie sicher, dass eine Zielfunktion deklariert ist. Inklusive der Anweisung, ob sie maximiert, beziehungsweise minimiert werden soll.
5. Abschließend muss nur noch der Solver aufgerufen und die Lösung ausgegeben werden.

4.2 Aufbau Datenbank

Das Herzstück aller itemspezifischer Informationen befindet sich in der Datenbank, welche aus einer Excel-Datei besteht. Darin enthalten sind 124 Items, mit allen wichtigen Eigenschaften, die aus dem Spiel Factory Town entsprungen sind. Ein kleiner Ausschnitt der Daten ist in der Tabelle 4.1 dargelegt.

`Herstellungszeit` (Unit/Sec) Gibt die Menge an, die in einer Sekunde vom Item hergestellt wird.

`Verkaufswert` (Coin/Unit) Bezeichnet den Gewinn der erzielt wird, wenn ein Item verkauft wurde. Zusätzlich wird hierbei zwischen dem Farbtönen der Münzen unterschieden. Ob ein Item verkauft werden kann hängt von der Verbrauchszeit ab, dementsprechend erst wenn es fertig konsumiert wurde.

`Verbrauchszeit` (Con Time) Beschreibt die Dauer wie lange ein Item pro Haus konsumiert wird. Erst nach Ablauf dieser Zeit kann das selbe Item erneut in Betracht genommen werden.

`Kategorie` (Category) Gliedert die Items in entsprechende Kategorien ein. Würde erst bei der Happiness ins Gewicht fallen.

`Zutaten` (Recipe) Definieren die Zusammensetzung eines Items, beziehungsweise welche Items, in welcher Menge, benötigt werden um ein Item herzustellen.

Item	$\frac{\text{Unit}}{\text{Sec}}$	Coin/Unit	Con Time	Category	Recipe
Grain	0.5	1	15	Basic Food	
Flour	0.25	4	30	Basic Food	Grain, Grain, Grain
Animal Feed	0.5				Grain, Grain
Bread	0.25	12	45	Basic Food	Flour, Flour, Fuel

Tabelle 4.1: Kleiner Bestandteil an Daten, die aus der Datenbank stammen.

Das Item „Grain“ symbolisiert die zuvor erwähnte natürliche Ressource, welche als Atom gilt und eine Basis legt. Natürliche Ressourcen lassen sich daran erkennen, dass sie keine Anmerkungen über die Zutaten haben.
Im Gegenteil dazu stehen Items, die aus einem, oder mehreren anderen Items bestehen. Beispielsweise ist das Item „Bread“, aus zweimal „Flour“ und einmal „Fuel“ zusammengesetzt. Da sich bei „Flour“ um kein atomares Item handelt, so muss bei der Herstellung von „Bread“ auch „Grain“ in betragt gezogen werden.
Gesondert sollte das Item „Fuel“ wahrgenommen werde. Da Fuel ausschließlich ein Platzhalter für eine der möglichen Ressource[10] ist, welche durch den User spezifiziert wird. Demgemäß muss sich der User, am Anfang der Benutzung vom Tool, für eine Kraftstoffart entscheiden. Angenommen „Wood“ wurde ausgewählt, so würde bei der Herstellung von „Bread“ ein halbes Stück „Wood“ ausreichen, um die benötigte Kraftstoffmenge zu decken.
Des weiteren wäre erwähnenswert, dass Items existieren die nicht verkauft werden können. Infolgedessen können sie weder konsumiert werden, noch sind sie einer Kategorie zugewiesen. Diese Besonderheit lässt sich anhand vom „Animal Feed“ veranschaulichen.

4.3 LP & IP

Im Tool wurden unterschiedliche Fragestellungen angegangen, welche dem User helfen sollen sein Town, beziehungsweise seine Spielart zu optimieren. Dabei ist es möglich zwischen grundlegenden Problemen, die eine Basis bilden, sowie anspruchsvolleren Aufgaben zu selektieren. Jeder dieser Fragestellung wurde vorerst als lineares Problem angegangen, lediglich durch kleine Modifikationen, lassen sich diese auch als integer Probleme darlegen. Der Aufbau der Formeln, sowie der Algorithmen ist übereinstimmend mit der Vorgehensweise, die im Abschnitt 4.1.3 beschrieben wurde. Variablen die für das LP \ IP relevant sind, werden durch die Schriftdicke hervorgehoben.

[10] Eine Einheit fuel kann mit 1Fertilizer, $\frac{1}{2}$Wood, $\frac{1}{4}$Coal, oder $\frac{1}{8}$Magma hergestellt werden.

4.3.1 Arbeitsverteilung für Items

Die hier zugrundeliegend Frage handelt, davon wie man seine Arbeitskräfte optimal verteilt, um die Herstellungsmenge eines Items zu maximieren. Dem Benutzer steht vier Einstellmöglichkeit[11] zur Verfügung. Primär ob das Problem linear, oder ganzzahlig gelöst werden soll. Zudem ob die Einschränkung der Verbrauchszeit ein Faktor spielt. Sollte das der Fall sein, so setzt sich die Limitierung für ein Item aus, der $\frac{\text{Anzahl der Häuser}}{\text{Itemabhängige Verbrauchszeit}}$, zusammen.

Formel Um eine Formel aufzustellen müssen zuerst die Variablen deklariert werden, welche aus zwei separaten mit nicht negativen Einträgen versehenen Vektoren **v, vPerSec** bestehen. Abhängig von der Entscheidung des Benutzers(linear\ integer) werden reelle Zahlen, oder eine Kombination verwendet.

$$\{\mathbf{v},\ \mathbf{vPerSec} \in \mathbb{R}^n\}, \text{ oder } \{\mathbf{v} \in \mathbb{Z}^n \text{ und } \mathbf{vPerSec} \in \mathbb{R}^n\}$$

Der Vektor **vPerSec** ist ein Faktor für die Herstellungszeit, wohingegen **v** die Verteilung der Arbeiter repräsentiert. Die Größe n entspricht der Anzahl an involvierten Items. Konkret sind die Einträge der Vektoren sinnbildlich für bestimmte Items, welche aus der Vereinigung aller Zutaten generiert werden. Wohlgemerkt, dass die selbe Zutat aus zwei unterschiedlichen Rezepten nicht gleichgestellt ist. Verständlicher abgebildet ist es anhand von der Ausgabe 4.3.1.
Folglich werden die Restriktionen angegangen, beginnend mit der Arbeiter-Einschränkung. Die Summe aller Elemente von **v** darf nicht größer sein, als die Konstante $a \in \mathbb{Z}$, die sich aus der Anzahl Arbeitern ergibt. Die Konstante a wird aus der Benutzereingabe berechnet, dabei übergibt der Benutzer einerseits das Level der Häuser, anderseits deren Anzahl. Kombiniert ist $a = (\text{ Level der Häuser} \cdot 2) \cdot \text{Anzahl der Häuser} + 4$.

$$\sum_{i=0}^{n} \mathbf{v_i} \leq a$$

Zudem ist eine Einschränkung der Herstellungszeit eingebunden, die den Faktor Zeit miteinbezieht. Dafür wird ein neuer Koeffizientenvektor $z \in \mathbb{R}^n$ integriert, welcher die Herstellungszeit der einzelnen Item enthält. Dieser wird zeilenweise mit **v** multipliziert und muss bei einem linear Problem gleich, sonst größer gleich **vPerSec** sein.

$$\{\mathbf{vPerSec} = \mathbf{v} \circ z\}, \text{ oder } \{\mathbf{vPerSec} \leq \mathbf{v} \circ z\}$$

Gemeinsam mit der vorherigen Einschränkung bildet die Rezept-Restriktion die Hauptsäule der Dosierung. Dabei besitzt jedes nicht atomare Item $i \in \mathbf{vPerSec}$ eine eigene Menge R_i, welche eine Zusammensetzung der benötigten Zutaten $r \in R_i$ widerspiegelt. Deckungsgleich referenziert r auf die entsprechend dazugehörige Variable in **vPerSec**. Zusätzlich existiert für jede Zutat r eine bindende Verteilungskonstante $vk_{i,r} \in \mathbb{Z}$, die besagt wie oft die Zutat im Rezept vorkommt. Ableitend zur Einschränkung der Herstellungszeit gilt bei einem linear Problem gleich, sonst kleiner gleich i.

[11] Die vier Varianten im Überblick: lineares Problem, integer Problem, lineares Problem mit der Einschränkung der Verbrauchszeit, integer Problem mit der Einschränkung der Verbrauchszeit.

$$\{\forall \mathbf{i} \in \mathbf{vPerSec}\ \forall \mathbf{r} \in \mathbf{R_i} \text{ gilt: } \mathbf{i} = \frac{\mathbf{r}}{vk_{i,r}}\}, \text{ oder}$$
$$\{\forall \mathbf{i} \in \mathbf{vPerSec}\ \forall \mathbf{r} \in \mathbf{R_i} \text{ gilt: } \mathbf{i} \leq \frac{\mathbf{r}}{vk_{i,r}}\}$$

Abhängig von der getroffen Einstellung des Benutzers kann adaptiv gewählt werden, ob die Einschränkung der Verbrauchszeit miteinbezogen werden soll. Für diesen Zweck wird die Variable **itemToMax** $\in$ **vPerSec**, für die die Rechnung durchgeführt wird hervorgehebt. Demgemäß symbolisiert **itemToMax** die hergestellte Menge pro Sekunde des zu maximierendem Items, somit darf sie nicht größer sein als die Konsum-Limitierung. Zur Vervollständigung bezeichnen die Konstanten $g, c_{itemToMax} \in \mathbb{Z}$ einerseits die Anzahl an Gebäuden, andererseits die Verbrauchszeit.

$$\mathbf{itemToMax} \leq \frac{g}{c_{itemToMax}}$$

Abschließend muss nur noch die Variable **itemToMax** maximiert werden.

$$\max\ \mathbf{itemToMax}$$

Zusammengesetzt ergibt das lineare Problem mit Berücksichtigung der Verbrauchszeit folgende Konstellation:

$$\begin{array}{rc} \max & \mathbf{itemToMax} \\ \mathbf{v}, \mathbf{vPerSec} & \in \mathbb{R} \\ \mathbf{v}, \mathbf{vPerSec} & \geq 0 \\ \sum_{i=0}^{n} \mathbf{v_i} & \leq a \\ \mathbf{vPerSec} & = \mathbf{v} \circ z \\ \forall \mathbf{i} \in \mathbf{vPerSec}\ \forall \mathbf{r} \in \mathbf{R_i} \text{ gilt: } \mathbf{i} & = \frac{\mathbf{r}}{vk_{i,r}} \\ \mathbf{itemToMax} & \leq \frac{g}{c_{itemToMax}} \end{array} \tag{4.1}$$

Ausgabe Die Ausgabe des linearen Problems 4.1 liefert für das zu maximierende Item „Bread“ folgende Ausgabe:

$$\begin{array}{rc} \text{Objective value} & = 0.\overline{1} \\ \text{Bread} & = 0.\overline{4} \\ \text{Bread} \leftarrow \text{Flour} & = 0.\overline{8} \\ \text{Bread} \leftarrow \text{Flour} \leftarrow \text{Grain} & = 1.\overline{3} \\ \text{Bread} \leftarrow \text{Flour} & = 0.\overline{2} \end{array}$$

Die präsentierten Werte sind in **v** enthalten und legen die Verteilung der Arbeitskräfte dar. Lediglich verkörpert „Objective value“ den maximalen **itemToMax** Wert.
Wobei sich der User-Input auf

- Specify House lv.: 1
- Specify Nr. of buildings: 5

- Specify Fuel Type: Wood

belaufen hat. Die übrigen Information wie die Rezepte, oder die Verbrauchszeit & Herstellungszeit der jeweiligen Items können der Tabelle 4.1 entnommen werden.

4.3.2 Ranking vom Verkaufswert

Diese Fragestellung basiert auf den Ergebnissen der vorherigen Frage und soll Auskunft geben, welche Item's sich besonders in der Herstellung rentieren. Dabei haben zwei Faktoren ausschlaggebenden Einfluss, einmal wie viel Münzen pro Sekunde für ein bestimmtes Item erzielt werden können, sowie die benötige Anzahl an involvierten Arbeitern.
Analog zur vorherigen Fragestellung, kann der Benutzer auch hier zwischen den vier Varianten wählen, allerdings bezieht sich die Einstellung auf die Struktur der Gleichungen 4.1 und nicht auf die der Gleichungen 4.2.

Formel Resultierend aus der Vorarbeit, der Gleichungen 4.1, erhalten wir für jedes $item \in itemList$ die maximale $\mathbf{mPerSec_{item}} = \frac{\text{hergestellte Menge}}{\text{Sekunde}}$, sowie die Verteilung der Arbeitskräfte $\mathbf{v_{item}}$. Die Menge $itemList$ enthält alle $item$, die in der Datenbank eingetragen wurden. Zudem repräsentiert die Summe aller Einträge $\mathbf{v_{item}}$ die Anzahl eingesetzten Arbeitern $w_{item} \in \mathbb{R}$.

$$\forall item \in itemList \sum_{i=0}^{n} \mathbf{v_{item,i}} = w_{item}$$

Im letzten Schritt muss nur noch der itemspezifische Verkaufswert-Koeffizient $k_{item} \in \mathbb{Z}$ eingebunden werden, um den Richtwert $rw_{item} \in \mathbb{R}$ zu bestimmen.

$$\forall item \in itemList \text{ gilt: } \frac{\mathbf{mPerSec_{item}} \cdot k_{item}}{w_{item}} = rw_{item}$$

Das Schema kann dieserart hergeleitet werden:

$$\begin{aligned} \forall item \in itemList \textstyle\sum_{i=0}^{n} \mathbf{v_{item,i}} &= w_{item} \\ \forall item \in itemList \text{ gilt: } \tfrac{\mathbf{mPerSec_{item}} \cdot k_{item}}{w_{item}} &= rw_{item} \end{aligned} \tag{4.2}$$

Ausgabe Ausgehend von der selben Ausgangslage, wie bei der Ausgabe 4.3.1, wird unter anderem das Ranking für die blaue Münzen 4.2 erstellt. Die übrigen Ranking's der gelben, roten, violetten Münzen werden nicht dargestellt, sind jedoch identisch aufgebaut. Damit die relevanten Information auf den ersten Blick zu sehen sind, werden die Richtwerte absteigend sortiert und nach der Münzfarbe getrennt.

4.3.3 Höchstmöglicher Profit einer Münzenart

Sollte es der Fall sein, dass ein Ranking vorhanden ist, so möchte man bestimmt wissen inwiefern man eine konkrete Münzfarbe maximieren kann. Diese Frage wird in diesem Abschnitt angegangen. Auch hierbei steht dem Benutzer frei zu Auswahl, ob er das Problem linear, oder mit ganzzahligen Werten angeht. Der Unterschied hierbei ist, dass die Einschränkung

Nr.	Item	$\frac{\text{hergestellte Menge}}{\text{pro Sekunde}}$	$\frac{\text{Münze}}{\text{Teil}}$	$\frac{\text{Münzen}}{\text{Sek.}}$	Anz. Arbeiter	Richtwert
1	Remedy	$0.1\overline{6}$	4	$0.\overline{6}$	$2.\overline{3}$	0.2857...
2	Antidote	$0.08\overline{3}$	14	$1.1\overline{6}$	$5.4\overline{9}$	$0.\overline{21}$
3	Medical Wrap	$0.0\overline{5}$	18	1.0	$5.8\overline{3}$	0.1714...
4	Ointment	$0.0\overline{5}$	10	$0.\overline{5}$	$3.\overline{5}$	0.1562...
5	Poultice	$0.08\overline{3}$	3	0.25	$1.91\overline{6}$	0.1304...
6	Water Ether	$0.\overline{1}$	6	$0.\overline{6}$	$5.\overline{2}$	0.1276...
7	Protein Shake	$0.0\overline{5}$	4	$0.\overline{2}$	$1.8\overline{3}$	$0.\overline{12}$
8	Fish Oil	$0.\overline{1}$	2	$0.\overline{2}$	$2.\overline{4}$	$0.\overline{09}$
9	Bandage	$0.\overline{1}$	1	$0.\overline{1}$	$1.\overline{4}$	0.0769...
10	Elixir	0.0128...	32	0.4113...	$13.\overline{9}$	0.0293...
11	Water Ring	0.0132...	30	0.3984...	14.0	0.0284...
12	Cure Spellbook	0.0133...	20	0.2664...	14.0	0.0190...

Tabelle 4.2: Ranking der blauen Münzen als lineare Problem, mit Beachtung der Verbrauchszeit.

der Verbrauchszeit von vornherein angenommen wird. Würde man die Restriktion außer Acht lassen, so wäre die Lösung der erste Eintrag des Ranking's.

Formel Ausgangspunkt der Formel ist eine fertiges Ranking für eine bestimmte Münzfarbe $color \in \{gelb, rot, blau, violett\}$, die der Benutzer spezifizieren hat. Deswegen sind die einzelnen $item \in ItemList_{color}$, unterdessen nach der Münzfarbe selektiert worden. Zudem sind itemspezifische Werte wie $mPerSec_{item}$, k_{item}, w_{item}, oder auch die Anzahl Arbeitern a bekannt. Die ursprüngliche LP Variable $mPerSec_{item}$ wird hier nicht hervorgehoben, da sie bereits durch das vorherigen LP gelöst wurde. Dementsprechend repräsentiert sie, in dem aktuellen Problem, nur eine Konstante. Zuallererst müssen für das Problem (linear\ integer) Variablen deklariert werden, welche aus einem mit nicht negativen Einträgen versehenen Vektor $\mathbf{t}$ bestehen.

$$\{\mathbf{t} \in \mathbb{R}^m\}, \text{ oder } \{\mathbf{t} \in \mathbb{Z}^m\}$$

Die Größe des Vektors $\mathbf{t}$ hängt von der Mächtigkeit der Menge $ItemList_{color}$ ab, ergänzend gibt es eine klare Zuweisung der Einträge von $\mathbf{t}$ und $item$'s. Zudem ist es entscheidend, dass es sich hierbei um Entscheidungsvariablen handelt, die einen Wertebereich von $\mathbb{W}_t = [0, 1]$ besitzen. Infolgedessen kann bestimmt werden ob sich die Herstellung von einem Item in dieser Kombination lohnt, oder doch eine andere besser wäre. Falls $\mathbf{t} \in \mathbb{R}^m$ so kann der Variablenwert, als Prozentsatz gesehen werden. Äquivalent zur Formel 4.3.1 besitzt auch dieses Problem eine Arbeiter-Restriktion.

$$\sum_{item}^{itemList_{color}} w_{item} \cdot \mathbf{t_{item}} \leq a$$

Zu guter Letzt muss die Zielfunktion definiert werden. Dabei wird die bestmögliche Verteilung, von den am meisten rentierenden Items, gesucht.

$$\max \sum_{item}^{itemList_{color}} k_{item} \cdot mPerSec_{item} \cdot \mathbf{t_{item}}$$

Zusammengefasst entspringt für dieses lineares Problem folgende Formel:

$$\begin{gathered} \mathbf{t} \in \mathbb{R} \\ \mathbb{W}_t = [0,1] \\ \sum_{item}^{itemList_{color}} w_{item} \cdot \mathbf{t_{item}} \leq a \\ \max \sum_{item}^{itemList_{color}} k_{item} \cdot mPerSec_{item} \cdot \mathbf{t_{item}} \end{gathered} \tag{4.3}$$

Ausgabe Die betrachtete Problemstellung angewendet auf die Tabelle 4.2 ergibt, die in der Tabelle 4.3 dargestellte Ausgabe.

Item	$\frac{\text{Münzen}}{\text{Sek.}}$	Anz. Arbeiter	Faktor
Medical Wrap	1.0	$5.8\overline{3}$	1.0
Remedy	$0.\overline{6}$	$2.\overline{3}$	1.0
Ointment	0.0520...	$0.\overline{3}$	0.0937...
Antidote	$1.1\overline{6}$	$5.4\overline{9}$	1.0

Tabelle 4.3: Die bestmögliche Verteilung, von den am meisten rentierenden Items, der blauen Münzen.

4.3.4 Gewichtung der Münzenarten

Führt man die vorherige Idee weiter, so würde eine Gewichtung nach Münzfarbe definitiv ein übergeordnetes Problem charakterisieren. Derartig ist es dem Benutzer offen eine farbabhängige Verteilung $fv_{color} \in \mathbb{Z}^4$ zu definiere. Beispielsweise doppelt so viele blaue Münzen zu erwirtschaften wie rote. Nicht zu vernachlassen, dass man weiterhin den maximalen Gewinn erzielen möchte. Sollte eine Farbe nicht ins Gewicht fallen sollen, so wird sie null gesetzt. Die Problemstellung kann erneut sowohl linear, als auch ganzzahlig angegangen werde. Allein die Berücksichtigung der Verbrauchszeit wird vorausgesetzt.

Formel Da es sich hierbei um ein übergeordnetes Problem handelt, wird diese Formel viele Parallelen zu der Formel 4.3.3 haben. Dementsprechend existiert bereits eine Zuordnung aller *item* zu ihrer dazugehörigen Menge $itemList_{color}$. Die Zuteilung wird wiedermal von der Münzfarbe abhängig gemacht ($colorList = \{gelb, rot, blau, violett\}$ und $color \in colorList$). Mit der Ausnahme, dass die Menge $itemList_{color}$, repräsentativ für die Farben, viermal vorkommt. Darüber hinaus sind itemspezifische Werte wie $mPerSec_{item}$, k_{item}, w_{item}, oder die Anzahl Arbeitern a bereits vorgerechnet.
Adaptive zur Formel 4.3.3 verwenden lineare \ ganzzahlige Variablen deklariert, welche aus vier mit nicht negativen Einträgen versehenen Vektoren $\mathbf{t_{color}}$ bestehen.

$$\{\mathbf{t_{color}} \in \mathbb{R}^{|itemList_{color}|}\}, \text{ oder } \{\mathbf{t_{color}} \in \mathbb{Z}^{|itemList_{color}|}\}$$

Ergänzend sind die Einträge von $\mathbf{t_{color}}$ Entscheidungsvariablen, welche ein klare Zuweisung zu *item*'s haben.

$$\mathbb{W}_{tcolor} = [0,1]$$

Falls $\mathbf{t_{color}} \in \mathbb{R}^{|itemList_{color}|}$ so kann der Variablenwert, als Prozentsatz gesehen werden. Weiter werden vier zusätzliche Variablen benötigt, die in einem mit nicht negativen Einträgen versehenen Vektor $\mathbf{o}$ enthalten sind. Die Einträge sind an *color* gebunden.

$$\{\mathbf{o_{color}} \in \mathbb{R}^4\}, \text{ oder } \{\mathbf{o_{color}} \in \mathbb{Z}^4\}$$

Außerdem operiert die Variablen in $\mathbf{o}$ auf folgenden Wertebereich:

$$\mathbb{W}_{ocolor} = [1, \infty)$$

Äquivalent zur Formel 4.3.1 besitzt auch dieses Problem eine Arbeiter-Restriktion.

$$\sum_{color}^{colorList} \sum_{item}^{itemList_{color}} w_{item} \cdot \mathbf{t_{color,item}} \leq a$$

Darüber hinaus soll für jede Farbe eine Summe berechnet werden, die sich aus einer Kombination von ausgewählten *item*, deren $\frac{\text{hergestellte Menge}}{\text{Sekunde}}$ und deren Verkaufswert zusammensetzt. Diese Summe soll im Umkehrschluss größer sein als die farbabhängige Limitierung.

$$\forall color \in colorList \text{ gilt: } \sum_{item}^{itemList_{color}} mPerSec_{item} \cdot k_{item} \cdot \mathbf{t_{color,item}} \geq \mathbf{o_{color}}$$

Die letzte Restriktion bildet die Koeffizient Verhältnisse zwischen den Farben. In anderen Worten gibt sie an, dass doppelt so viele blaue wie rote Münzen hergestellt werden sollen. Alle Farben die der Benutzer in seiner Verteilung fv mit null gekennzeichnet hat, sollen außer Acht gelassen werden.

$$\forall color1 \in colorList \; \forall color2 \in colorList, \text{ wobei } fv_{color1} \neq 0 \text{ und } fv_{color2} \neq 0 \text{ gilt:}$$
$$\mathbf{o_{color1}} \cdot fv_{color2} = \mathbf{o_{color2}} \cdot fv_{color1}$$

Final wird die Zielfunktion solchermaßen angegeben:

$$\max \sum_{color}^{colorList} \mathbf{o_{color}}$$

Alle Terme vereint generieren das unten abgebildete lineare Problem.

$$\begin{gathered}
\max \textstyle\sum_{color}^{colorList} \mathbf{o_{color}} \\
\mathbf{t_{color}} \in \mathbb{R}^{|itemList_{color}|} \\
\mathbb{W}_{tcolor} = [0, 1] \\
\mathbf{o_{color}} \in \mathbb{R}^4 \\
\mathbb{W}_{ocolor} = [1, \infty) \\
\textstyle\sum_{color}^{colorList} \sum_{item}^{itemList_{color}} w_{item} \cdot \mathbf{t_{color,item}} \leq a \\
\forall color \in colorList \text{ gilt: } \textstyle\sum_{item}^{itemList_{color}} mPerSec_{item} \cdot k_{item} \cdot \mathbf{t_{color,item}} \geq \mathbf{o_{color}} \\
\forall color1 \in colorList \; \forall color2 \in colorList, \text{ wobei } fv_{color1} \neq 0 \text{ und } fv_{color2} \neq 0 \text{ gilt:} \\
\mathbf{o_{color1}} \cdot fv_{color2} = \mathbf{o_{color2}} \cdot fv_{color1}
\end{gathered} \tag{4.4}$$

Ausgabe Fortgehend von der selben Benutzereingabe wie in der Ausgabe 4.3.1, wird zusätzlich mit der Ergänzung der Münzfarb-Verteilung

- Gewichtung der gelben Münzen = 0
- Gewichtung der rot Münzen = 1
- Gewichtung der blau Münzen = 2
- Gewichtung der violett Münzen = 0

die in der Tabelle 4.4 dargestellte Ausgabe generiert.

Münzfarbe	**Item**	$\frac{\textbf{Münzen}}{\textbf{Sek.}}$	**Anz. Arbeiter**	**Faktor**
red	Reinforced Plank	0.5235...	1.3088...	0.3490...
red	Shirt	$0.\overline{6}$	$1.\overline{6}$	1.0
blue	Medical Wrap	0.5470...	3.1911...	0.5470...
blue	Remedy	$0.\overline{6}$	$2.\overline{3}$	1.0
blue	Antidote	$1.1\overline{6}$	$5.4\overline{9}$	1.0

Tabelle 4.4: Der maximale Gewinn an Münzen unter der Bedingung, dass doppelt so viele blaue wie rote Münzen erwirtschaftet werden sollen.

5

Gegenüberstellung der Resultate LP & IP

Ziel dieses Kapitels ist es eine Gegenüberstellung zwischen den LP und den IP Resultaten zu liefern. Grundlegend kann die im letzten Abschnitt von Integer Optimierung 2.3 getroffene Aussage, dass die LP Lösungen mindestens genau so gut oder sogar besser sind als die Lösungen eines IP, als Basis genommen werden.
Die für diesen Vergleich genutzte Plattform, verwendet den in der Ausgabe 4.3.1 definierten User-Input. Um einen sauberen Vergleich präsentieren zu können, wurden Abbildungen von der Fragestellung 4.3.2 erstellt. Dies hat sich besonders angeboten, da für alle 124 Items (x-Achse) eine Arbeitsverteilung, welche im Unterkapitel 4.3.1 beschrieben ist, berechnet wurde. Ansonsten wurden alle vier Varianten der Fragestellung verwendet, wobei immer zwei gegenübergestellt wurden. Die Abbildung 5.1 repräsentiert die Kombination LP(rote Punkte) gegen IP (blaue Punkte) ohne Verbrauchszeit, wohingegen die Abbildung 5.2 die Verbrauchszeit miteinbezieht. Der dargestellte Wert(y-Achse) beschreibt die Anzahl Münzen, die pro Sekunde für ein bestimmtes Item erwirtschaftet werden können, im Verhältnis zu der Anzahl an investieren Arbeiter.

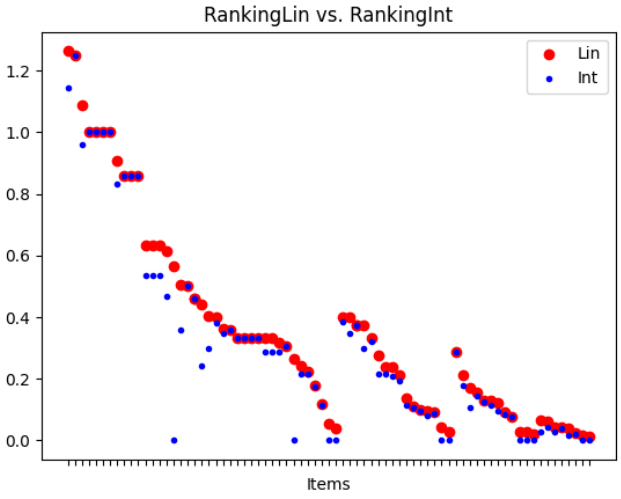

Abbildung 5.1: Gegenüberstellung linear gegen integer Programming.

Auf den ersten Blick lässt sich erkennen, dass in keiner der beiden Abbildungen eine integer Lösung besser ist als eine lineare Lösung. Die im Schaubild 5.1 gekennzeichneten integer Werte sind in den meisten Fällen sehr Nah oder gleich, aber nie besser. Der Grund dafür ist der Mangel der Verbrauchszeit-Restriktion, denn dadurch lassen sich alle Arbeiter in die Produktion integrieren, um die maximale Herstellungsmenge zu fabrizieren. Die Differenzen ist infolgedessen allein auf die nicht perfekt bilanzierte Arbeitsverteilung zurückverfolgen.
Manche ganzzahligen Resultate erhalten als optimale Lösung null. Ursache dafür ist, dass aufgrund der fehlender Arbeiter nicht alle Variablen gedeckt werden können. Beispielsweise würde das Item „Sandwich“ mindestens 16 Arbeiter benötigen.

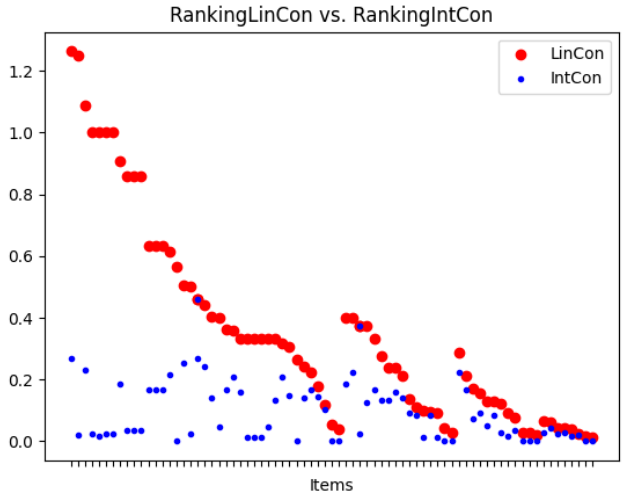

Abbildung 5.2: Gegenüberstellung linear gegen integer Programming mit Berücksichtigung der Verbrauchszeit.

Im Vergleich beider Abbildungen stich ein Unterschied wesentlich hervor. Die in der Abbildung 5.2 dargestellten integer Werte sind deutlich geringer, wobei die linearen Werte sich nicht verändert haben. Insofern ist auch die Differenz deutlich höher, die aus zwei Faktoren resultiert. Einerseits aus der zuvor erwähnten nicht perfekt bilanzierte Arbeitsverteilung und anderseits werden bei einem IP mehr Arbeiter involviert, um auf die selbe, durch die Verbrauchszeit-Bedingung gegebene, Herstellungsmenge zu kommen. Dementsprechend ist auch der zusammengesetzte Richtwert viel geringer.
Dem LP ist es hingegen möglich das Verhältnis an investierten Arbeitskräften zur gewünschten Herstellungsmenge linear zu senken.

6
Fazit

Zusammenfassend lässt sich mit Sicherheit sagen, dass LP und IP einen tiefgründigen Bereich der Optimierung repräsentieren. Sie ermöglichen eine Menge an abwechslungsreichen Fragestellungen, somit sind der Kreativität kaum Restriktionen gesetzt.
Wohingegen IP den praktischen Umfang an Modellierungsmöglichkeiten erhöht, indem Arbeiter nicht in zwei geteilt werden müssen, sind die LP Lösungen mindestens genauso gut oder sogar besser als die Lösung des IP.
Des Weiteren darf nicht vernachlässigt werden, dass LP in polynomialer Zeit berechnet werden kann, während IP ein NP-schweres Problem darstellt. Unter Betrachtung der Tatsache, dass ganzzahlige Probleme insofern als lineare Problem beschrieben werden können, wenn die Variablen $x \in \mathbb{Z}$ sind.
Zudem konnten mithilfe des OR-Tools konkrete Fragestellungen beantwortet werden, welche durch das einheitliche Vorgehen eine adaptive Struktur bilden konnten.
Ansonsten konnte das Spiel Factory Town mit seiner Komplexität faszinieren. Es bot eine breite Basis an durchführbaren Fragestellungen und eröffnete dementsprechend einen großen Spielraum. Die etliche Auswahl an unterschiedlichen Items macht es zusätzlich beschränkt umsetzbar, ohne Anwendung von LP\ IP, eine optimale Lösung zu finden.
Allem zusammen hab alle Faktoren zu einem exzellenten Gesamtpaket beigetragen, um ein intuitives Tool zu entwickeln, dass dem Benutzer verhelfen soll seinen Spielstil zu optimieren und sein Erfolg maximieren.

Literaturverzeichnis

[1] Solving an LP Problem. URL https://developers.google.com/optimization/lp/lp_example. Besucht: 13.08.2021.

[2] Solving Constraint Integer Programs. URL https://www.scipopt.org/. Besucht: 13.08.2021.

[3] What is integer programming? URL https://www.ibm.com/docs/en/icos/12.8.0.0?topic=problem-what-is-integer-programming. Besucht: 11.08.2021.

[4] About OR-Tools. URL https://developers.google.com/optimization. Besucht: 12.08.2021.

[5] Decision variables. URL https://www.ibm.com/docs/en/icos/12.9.0?topic=types-decision-variables. Besucht: 11.08.2021.

[6] Lineare Optimierung. URL https://de.wikipedia.org/wiki/Lineare_Optimierung. Besucht: 11.08.2021.

[7] Mixed-Integer Programming (MIP) – A Primer on the Basics. URL https://www.gurobi.com/resource/mip-basics/. Besucht: 12.08.2021.

[8] Ganzzahlige lineare Optimierung. URL https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung. Besucht: 12.08.2021.

[9] Erik Asmussen. Factory Town, 2019. URL https://store.steampowered.com/app/860890/Factory_Town/. Besucht: 10.08.2021.

[10] Prof. Dr. Shuchi Chawla. LP Relaxation and Rounding. *University of Wisconsin-Madison*, page 2, 2009.

[11] Saul I. Gass. George B. Dantzig. *Springer*, 147:217–240, 2011.

[12] Prof. Dr. Anupam Gupta. LPs: Algebraic View. *Carnegie Mellon University*, page 2, 2011.

[13] L. V. KANTOROVICH. MATHEMATICAL METHODS OF ORGANIZING AND PLANNING PRODUCTION (reprint). *Operations Research*, 11(6):863–888, 1963.

[14] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[15] Prof. Dr. Marco Lübbecke. lineare Optimierung. URL https://wirtschaftslexikon.gabler.de/definition/lineare-optimierung-39312. Besucht: 12.08.2021.

[16] Heiner Müller-Merbach. Operations Research. *Verlag Franz Vahlen, München*, 3:89, 1973.

[17] George J. Stigler. The Cost of Subsistence. *Journal of Farm Economics*, 27(2):303–314, 1945.

[18] Prof. Dr. Uwe Suhl. Lineare Optimierung, 2020. URL https://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/technologien-methoden/Operations-Research/Mathematische-Optimierung/Lineare-Optimierung. Besucht: 11.08.2021.

Declaration on Scientific Integrity
Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor
Mateusz Palasz

Matriculation number — Matrikelnummer
2017-058-777

Title of work — Titel der Arbeit
Optimierungsanfragen für Town Building Spiele mithilfe von LP- & IP-Solver

Type of work — Typ der Arbeit
Bachelorarbeit

Declaration — Erklärung
I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, 16.08.2021

Signature — Unterschrift