*Technische Dokumentation*

**Fahrzeugbedarfsplanung**

**Version 4.0**

*Erstellt von:*

Roth, Felix – 289 043

Beck, Julian – 286 876

am 20. Januar 2015

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc441077191)

[2. Intention und Anwendungsbereich der Software 3](#_Toc441077192)

[2.1. Intention 3](#_Toc441077193)

[2.2. Versionshistorie 4](#_Toc441077194)

[2.2.1. Über Version 1.0 4](#_Toc441077195)

[2.2.2. Über Version 2.0 4](#_Toc441077196)

[2.2.3. Über Version 3.0 5](#_Toc441077197)

[2.3. Unsere Weiterentwicklung zu Version 4.0 5](#_Toc441077198)

[2.4. Mögliche Weiterentwicklungen 6](#_Toc441077199)

[3. Vorstellung der Version 4.0 7](#_Toc441077200)

[3.1. Bestehende Funktionen 8](#_Toc441077201)

[3.1.1. Bedarf berechnen 8](#_Toc441077202)

[3.1.2. Optimum berechnen 8](#_Toc441077203)

[4. Der Ansatz 10](#_Toc441077204)

[4.1. Busnummernzuweisung 10](#_Toc441077205)

[4.2. Tourverschiebung 13](#_Toc441077206)

[Quellenverzeichnis 15](#_Toc441077207)

[Abbildungsverzeichnis 16](#_Toc441077208)

# Einleitung

Im Rahmen der Veranstaltung „Anwendung der linearen Optimierung“ kam uns die Aufgabe zu, die Software „Fahrzeugbedarfsplanung“ zu analysieren und zu verbessern.

Zu Beginn der Arbeit lag uns Version 3.0, datiert auf den 15.07.2005 vor.  
Diese Version wurde durch Elena Steingrad und Karolin Krieg kompiliert.

Die weitere Versionshistorie ist wie folgt:

Version 2.0 - Wintersemester 2000/2001  
Holger Kleinbub und Felix Soremski

Version 1.1 - Sommersmester 1996  
Reinhard Bär, Ahmad Marvani und A.-N. Tulug Yücel

Version 1.0 - Wintersemester 1995/1996  
Silke Westermann und Jutta Rohmann

Die letzte Dokumentation liegt für Version 2.0 vor, datiert auf den 25.01.2001.  
Für Version 2.0 wurde die Software komplett neu erstellt.

Auf alle Änderungen gegenüber Version 2.0 gehen wir im Folgenden ein.

Für die Erklärung der grundlegenden Funktionen der Software zitieren wir in großem Umfang die Dokumentation der Fahrzeugbedarfsplanung für Version 2.0 von Holger Kleinbub und Felix Soremski.

# Intention und Anwendungsbereich der Software

## **Intention**

Die betriebliche Systemforschung bietet die Möglichkeit, einen Ansatz zu formulieren, der den Einsatz von Fahrzeugen eines Transportunternehmens steuert und optimiert. Für solche Unternehmen sind die Gewinnmarchen äußerst gering und es bedarf einer genauen Analyse, wie mit den vorhandenen Produktionsmitteln wie Busse, Taxen oder LKW ein bestimmtes Tour-Programm gelöst werden kann und ob dann eventuell noch Ressourcen frei sind.  
In der Theorie spielt es keine Rolle, ob Speditionen ihre Lkw- Einsatzplanung betreiben oder Busunternehmen den Einsatz ihrer Busse minimieren. Diese Software bietet eine kleine Hilfe, um einen Einsatzplan zu erstellen, der Ausgangspunkt für Unternehmensentscheidungen ist. So kann erkannt werden, ob noch Ressourcen zur Verfügung stehen oder man kann auf Notfallsituationen wie z.B. einen Motorschaden angemessen reagieren.

## Versionshistorie

## Über Version 1.0

In der Methodenbank der Fachhochschule existierte bereits ein entsprechendes Tool 1.0, das dieses Problem auf eine einfache Art und Weise zu lösen versuchte. Es wurden darin aber lediglich die Touren pro Periode summiert und so der Fahrzeugbedarf errechnet. Ein LP-Ansatz mit Benutzung eines Solvers ist im Programm nicht vorhanden.

Aus der Ergebnisgrafik, die von der Software angezeigt wird, ist auch nicht ersichtlich, welche Tour welchem Fahrzeug zugeordnet wurde.

Das Programm wies außerdem einige Probleme und Fehler auf. So konnten z.B. die eingegebenen Touren nicht mehr verändert oder gelöscht werden. Des Weiteren gab es Probleme beim Laden und Erweitern der Fahrpläne. Als störend erweist sich auch, dass die Software viele spezielle DLLs, sowie eine Grafikkomponente erforderlich macht, die nicht auf jedem Rechner vorhanden war.

## Über Version 2.0

Die Aufgabe des Wintersemesters 2000/2001 war es, die Software dahingehend zu erweitern, dass zur Lösung des Problems ein Ansatz der Linearen Programmierung gewählt wird. Zur Berechnung eines solchen Ansatzes stand wiederum ein Solver zur Verfügung (XA), der eingebunden wurde.

Es wurde entschieden, eine neue Anwenderschnittstelle für dieses Problem zu schreiben, um auch anderen Gruppen die Möglichkeit zu bieten, diese Software zu erweitern.

Das eigentliche Problem war der Ansatz. Ziel war es nicht wie in der bisherigen Version, lediglich die maximal benötigte Anzahl an Bussen zu berechnen. Dazu benötigt man keinen LP-Ansatz. Ziel war es, einen konkreten Einsatzplan zu erstellen, der Auskunft darüber gibt, welche Tour mit welchem Bus gefahren wird.

Eine zusätzliche Anforderung war auch, in diesen Einsatzplan einzugreifen und ihn flexibler gestalten zu können. So wie beim Ansatz zur Personalplanung sollte es möglich sein, bei geringer Überschneidung von Touren eine zu verschieben, um somit die Gesamtzahl an Bussen zu verringern.

Die Software für Version 2.0 wurde mit C++ und der Entwicklungsumgebung Visual C++ entwickelt.

## Über Version 3.0

Elena Steingrad und Karolin Krieg entwickelten Version 2.0 in der Form weiter, dass nach dem „XA“ nun auch der Solver „MOPS“ angebunden wurde. Weiterhin wurde aufgrund von Inkompatibilitäten die Hilfe-Funktion entfernt.

## Unsere Weiterentwicklung zu Version 4.0

Das ursprüngliche Ziel unserer Weiterentwicklung laut Pflichtenheft war die Anzeige des LP-Ansatzes, einer Begrenzung der Mitfahrer (auf 50 Personen) sowie des Einbaus einer einstündigen fixen Pause zwischen Fahrten.

Für uns selbst legten wir weiterhin die folgenden Ziele fest: Schaffung der kompletten Kompatibilität zu Windows 10 und Code-Verbesserung.

Nach der ersten Analyse des Programmcodes fiel auf, dass eine Anzeige des LP-Ansatzes nicht zielführend ist (Begründung siehe Abschnitt 4).

Nach einer initialen Installation von Visual Studio Professional 2015 (bezogen über DreamSpark) war es uns möglich, den Quellcode der Visual Studio 2005 Version, der bis dahin auf dem Rechner von Frau Steingrad lief, entsprechend zu importieren.

Aufgrund diverser Inkompatibilitäten musste ein grundlegendes Refactoring durchgeführt werden, bei welchem mehrere 100 Zeilen Code entfernt wurden und zur Wahrung der Kompatibilität zu Windows 10 Code hinzugefügt werden musste – die komplette Solveranbindung musste neue Schleifenvariablen erhalten.

Die Begrenzung der Mitfahrer auf 50 Personen wurde anschließend von uns erfolgreich in Angriff genommen. Der Einbau der fixen Pause erschien uns aufgrund der Komplexität des LP-Ansatzes nicht realisierbar.

Nach Rücksprache mit Herrn Serkan Önnisan, wissenschaftlicher Mitarbeiter von Herrn Prof. Dr. Michael Grütz, wurde entschieden, dass zur Wahrung der Kompatibilität der Open-Source-Solver „LP-Solve“ hinzugefügt werden soll.

Unter erheblichem Aufwand wurde eine neue Solveranbindung für LP-Solve entwickelt. Dies schloss sowohl die Entwicklung der entsprechenden C++-Klassen ein, als auch die dynamische Erstellung von BATCH-Dateien für die Parametrisierung des Solvers. Weiterhin musste das Parsing der Solver-Lösung neu implementiert werden.

Festgestellt wurde auch, dass der XA-Solver, welcher noch implementiert war, nicht nutzbar war. Aufgrund der Abschaffung des 16-Bit-Subsystems seit Windows Vista war dieser nicht mehr lauffähig. Dieser wurde im Auswahlfeld ausgegraut und die Referenzen entsprechend entfernt.

## Mögliche Weiterentwicklungen

Für die weitere Nutzung der Ergebnisse aus der Fahrzeugbedarfsplanung wäre eine Integration in die Personalplanung denkbar. Hierbei könnte beispielsweise ein Schichtplan für Busfahrer erzeugt werden.

Für die Personalplanung könnte die Temp-Datei Mops.res bzw. lps.out weiterverwendet werden.

# Vorstellung der Version 4.0

Unser erstes persönliches Ziel der Schaffung der Kompatibilität unter Windows 10 wurde durch Anpassung der Grafik-Library und der Integration des neuen Solvers erreicht.

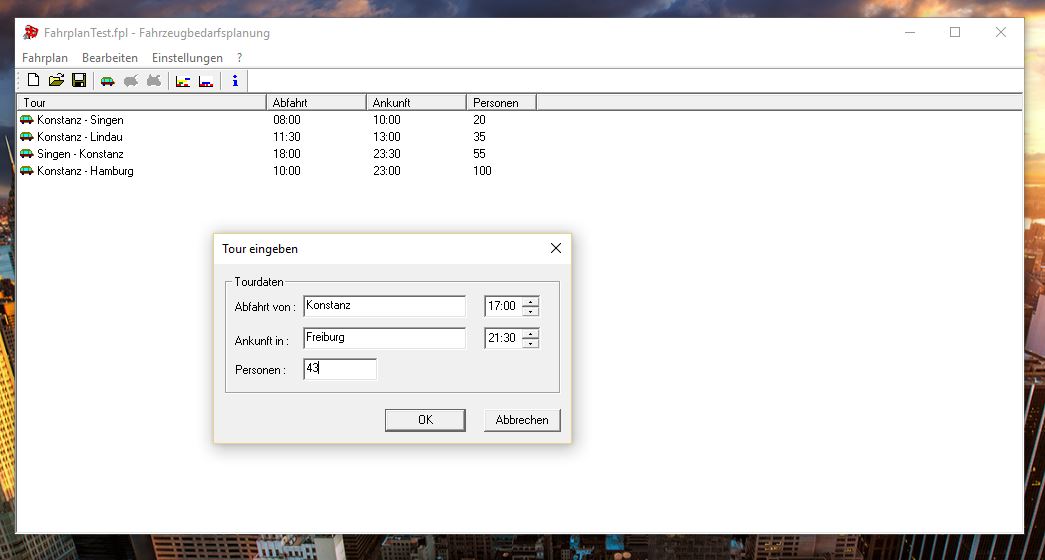


Abbildung 1 - Fahrzeugbedarfsplanung 4.0 nativ auf Windows 10 – Julian Beck

Das Ergebnis der Berechnung wird nach wie vor in Form einer Balkengrafik dargestellt. Das Zeichnen der Grafik wird vollständig vom Programm übernommen und ist per C++ realisiert.

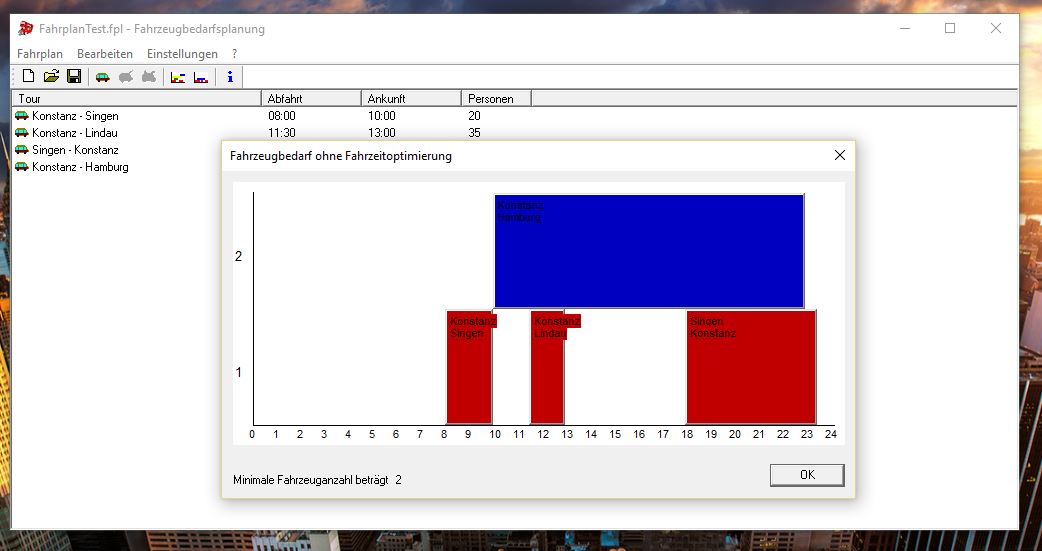


Abbildung 2 - Anzeige des Fahrzeugbedarfs – Julian Beck

In den Programmeinstellungen sind die Entfernung des XA und die Integration von „LP Solve“ ersichtlich. Die Nutzung von MOPS ist weiterhin möglich.

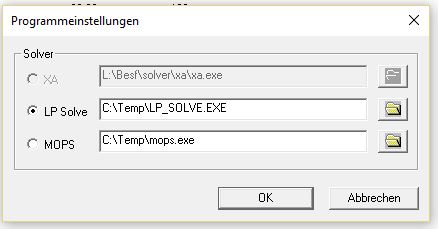


Abbildung 3 - der neue Solver "LP Solve" kann ausgewählt werden – Julian Beck

## Bestehende Funktionen

## Bedarf berechnen

Es wird der notwendige Fahrzeugbedarf berechnet, um den vorgegebenen Fahrplan zu bedienen. Zusätzlich werden die Touren den Fahrzeugen zugeordnet.

## Optimum berechnen

Bei dieser Art der Berechnung wird ein alternativer Fahrplan erstellt. Die Software versucht dabei die Abfahrts- bzw. Ankunftszeiten zu verschieben und somit den Fahrzeugbedarf zu minimieren.

Diese zwei Alternativen können gegenüber gestellt werden, um auf dieser Grundlage leichter Entscheidungen über den optimalen Fahrzeugeinsatz zu treffen.

In den folgenden Abbildungen wird ein Beispiel solcher Alternativberechnung demonstriert.

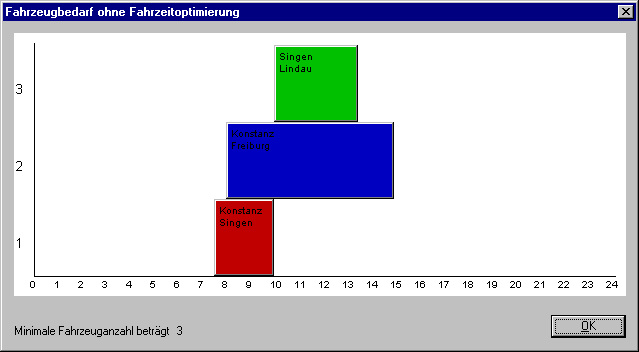


Abbildung 4 - ohne Optimierung (Abb. Version 2.0, Windows XP) - SoKl01

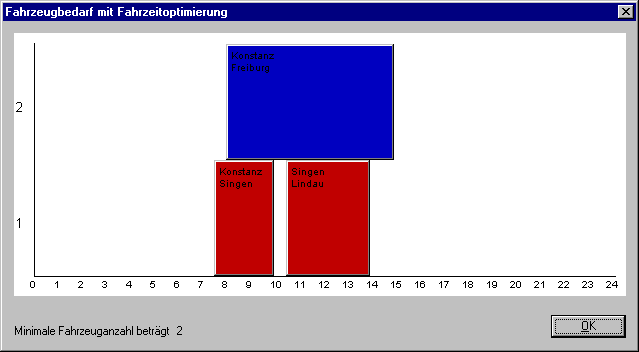


Abbildung 5 - mit Optimierung (Abb. Version 2.0, Windows XP) - SoKl01

# Der Ansatz

Die eigentliche Lösung erfolgt nicht mit einem LP-Ansatz. Es soll nun gezeigt werden, welche Möglichkeit existiert, um die Lineare Programmierung zu verwenden und wieso der Aufwand dafür aber zu hoch ist und es auch anders geht.

## Busnummernzuweisung

Wenn wir lediglich den Touren eine bestimmte Busnummer zuweisen wollen, ist das allgemein ein Ansatz zur Maschinenbelegung. Hierbei ist es wichtig, die Gleichungen korrekt zu formulieren. Abbildung 6 zeigt grafisch, wie bei der Maschinenbelegung der Ansatz formuliert werden müsste und welche Lösung erfolgen würde. In diesem Fall würden Hilfsvariablen in der Zielfunktion maximiert, die alles nach unten drücken.

Abbildung - Ansatz und Lösung mit Maschinenbelegung - SoKl01

Bus

Zeit

Bus

Zeit

Das Problem ist, dass bei Busfahrten natürlich die Tour nicht unterbrochen, bzw. mit einem anderen Bus fortgesetzt werden kann. Somit ergibt sich beim korrekten Aufstellen der Gleichungen der Effekt, dass die Touren künstlich verlängert werden müssten, um diese Verschiebung zu verhindern.

Abbildung - LP-Ansatz zur Fahrzeugplanung für 2 und 3 Touren - SoKl01

Busse

Zeit

Busse

Zeit

Dieses Problem lässt sich auch relativ einfach implementieren. Als Regel kann formuliert werden, dass eine Tour so lange verlängert wird, bis eine parallel laufende beendet wird, bzw. bis eine neue Tour beginnt. Diese Überprüfung muss für alle Touren erfolgen.

Durch die frühere Version bei der XA noch implementiert war stoßen unsere Vorgänger auf folgendes Problem: Der XA lässt lediglich 500 Variablen zu, dadurch wurde man gezwungen, die Zeitachse in ½-stündliche Intervalle zu unterteilen. Somit ergeben sich schon 48 Intervalle, die lediglich eine Planung von maximal 10 Bussen zulassen. Als Folge kann man am Ergebnis nur noch ablesen, welcher Bus zu welcher Periode eine Tour beginnt. Welche konkrete Tour das ist, weiß man nicht. Fangen zur selben Periode 2 Touren an, kann man sie nur noch über die Endzeit unterscheiden. Wurde eine dieser Touren aber künstlich verlängert, kann man nicht mehr nachvollziehen, in welcher Periode die eigentliche Endzeit dieser Tour wäre. Ein Vergleich ist somit nicht mehr möglich.

Genauso verhält es sich auch, wenn ein Bus zwei Touren direkt hintereinander fährt. Jetzt lässt sich nicht einmal sagen, wann die eine Tour endet und die nächste anfängt. Dieser Ansatz ist also nicht zu gebrauchen.

Auf diesem Weg kommt man aber dahin, dass man herausfindet, was zum Analysieren der Lösung eigentlich nur von Interesse ist. Das einzige, was interpretiert werden muss, sind die Anfangszeiten einer Tour. Lässt man die künstlichen Verlängerungen beiseite und die eigentlich nicht erlaubten Teilungen einer Tour auf einen unteren Bus zu, erhält man die Anfangszeiten und die Endzeiten spielen bei der Zuordnung von Busnummern zur Tour keine direkt Rolle mehr.

Abbildung - Schritte eines Algorithmus - SoKl01

Bus

Zeit

Bus

Zeit

Bus

Zeit

Bus

Zeit

Abbildung 7 zeigt das Vorgehen eines solchen Algorithmus, bei dem das Ende einer Tour in einer anderen Zeile liegen kann. Die 2 Grafiken zeigen die Lösung des XAs. Iteriert man von oben nach unten und links nach rechts durch die Ergebnismatrix, kommt man zu dem Punkt, bei dem ein Bus (in diesem Fall Bus 3) eine Tour beginnt. Diese Tour wird mit diesem Bus gefahren. Jetzt sucht man sich die konkreten Daten einer Tour, die in dieser Periode beginnt[[1]](#footnote-2). Diese Tour muss dann aus der Ergebnismatrix gelöscht werden. Danach kann man mit dem Iterieren fortfahren, bis wieder eine Tour beginnt. Diese Schritte werden wiederholt, bis die Matrix leer ist.[[2]](#footnote-3)

Es wurde gezeigt, dass für das Ergebnis die Perioden eine Rolle spielen, in denen Touren beginnen. Diese Perioden lassen sich aber auch anders erkennen. Zählt man pro Periode die Busse, die benötigt werden, erhält man einen Vector mit dem Bedarf.

Abbildung - Bedarfsvector - SoKl01

Bus

Zeit

1 1 1 1 1 1 2 2 3 3 3 3 3 3 2 1 1 1 1 1 1 0 0 0

Eine Tour beginnt immer dann, wenn sich die Zahl erhöht. Wendet man nun die gleiche Logik an wie zuvor beschrieben, kommt man zu dem gleichen Ergebnis. Jetzt ist der Algorithmus aber etwas einfacher, dadurch dass nur noch ein Vector und nicht mehr eine Matrix verwendet wird. Außerdem ist man unabhängig von der Anzahl an möglichen Variablen. Jetzt können beliebig viele Busse verwendet werden.

Um den Bedarf zu berechnen, kann man eine Schleife programmieren, die jede Tour durchläuft und den Vector hochzählt. Um es aber etwas komplizierter zu machen und auch einen Solver in unsere Anwendung zu integrieren, benutzen wir ihn als Zähler.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 x1 | + | 0 x2 | + | 0 x3 | + | 0 x4 | + | 0 x5 | - | x6 | = 0  x5  x5  x5  x5  x5 |
| 1 x1 | + | 0 x2 | + | 0 x3 | + | 0 x4 | + | 0 x5 | - | x7 | = 0 |
| 1 x1 | + | 1 x2 | + | 0 x3 | + | 0 x4 | + | 0 x5 | - | x8 | = 0 |
| 1 x1 | + | 1 x2 | + | 0 x3 | + | 1 x4 | + | 0 x5 | - | x9 | = 0 |
| 1 x1 | + | 1 x2 | + | 1 x3 | + | 1 x4 | + | 0 x5 | - | x10 | = 0 |
| 1 x1 | + | 1 x2 | + | 1 x3 | + | 1 x4 | + | 0 x5 | - | x11 | = 0 |
| 0 x1 | + | 1 x2 | + | 1 x3 | + | 1 x4 | + | 0 x5 | - | x12 | = 0 |
| 0 x1 | + | 1 x2 | + | 1 x3 | + | 1 x4 | + | 1 x5 | - | x13 | = 0 |
| 0 x1 | + | 0 x2 | + | 1 x3 | + | 1 x4 | + | 1 x5 | - | x14 | = 0 |
| 0 x1 | + | 0 x2 | + | 0 x3 | + | 1 x4 | + | 1 x5 | - | x15 | = 0 |
| 0 x1 | + | 0 x2 | + | 0 x3 | + | 0 x4 | + | 1 x5 | - | x16 | = 0 |
| 0 x1 | + | 0 x2 | + | 0 x | + | 0 x4 | + | 0 x5 | - | x17 | = 0 |
| x1 | = | 1 |  |  |  |  |  |  |  |  |  |
| x2 | = | 1 |  |  |  |  |  |  |  |  |  |
| x3 | = | 1 |  |  |  |  |  |  |  |  |  |
| x4 | = | 1 |  |  |  |  |  |  |  |  |  |
| x5 | = | 1 |  |  |  |  |  |  |  |  |  |

Abbildung 10 - Ansatz für Solver als Zähler - SoKl01

Die Zielfunktion kann vernachlässigt werden. Sie spielt keine Rolle. Als Lösung liefert der Solver in den Variablen x6 bis x6+48 die Anzahl an Bussen, die gerade benötigt werden.

## 4.2. Tourverschiebung

Wie in Kapitel 4.1 schon erwähnt, wurde versucht den Algorithmus dahingehend zu erweitern, dass er zeigt, welche Alternativen vorhanden sind, um Busse einzusparen. Gedacht war ein LP-Ansatz wie bei der Personalplanung. Es sollte zu einer Tour immer zwei Alternative geben, die jeweils eine halbe Stunde früher und später beginnen. Leider ist der Ansatz wie bei der Personalplanung nicht zu verwenden, weil bei jedem Ansatz in gewisser Weise einen Bedarf benötigt wird. Eine Tour, die eine ½ Stunde früher beginnt, deckt aber in einer Periode einen Bedarf, in der eigentlich gar keiner vorhanden ist.

Es wurden mehrere Ansätze versucht und einer war sogar dazu geeignet, Touren zu verschieben. Das Problem war aber, dass der Ansatz die Touren immer verschoben hat. Es konnte von unseren Vorgängern nicht erreicht werden, dass der Algorithmus, wenn es keine Möglichkeit zur Optimierung gab, die unveränderten Ausgangstouren zurück liefert.

Aber trotzdem kann man Entscheidungen treffen, die es ermöglichen, Tour Zeiten zu ändern und somit eventuell Busse einzusparen.

Abbildung - Beispiel für Tourverschiebung - SoKl01

Bus

Zeit

Bedarf

0

1

2

1

1

0

0

0

0

Schaut man sich wieder den Bedarf an, so kann gesagt werden, dass eine Tour in einer Periode beginnt und in der nächsten die Anzahl an benutzten Bussen niedriger ist, diese um eine Periode nach hinten verschoben werden kann.

Diese kleine Erweiterung ist einfach in den Algorithmus integrierbar und funktioniert für Verschiebungen von einer halben Stunde. Wesentlich mehr Aufwand wäre nötig, wenn man zulassen wollte, gleichzeitig zwei Touren (nach vorn und nach hinten) zu verschieben. Dann käme wohl für Informatiker eher der größte Konkurrent zu einem LP-Ansatzes in Frage: Die Rekursion.

# Quellenverzeichnis

Wie bereits erwähnt, beziehen wir uns ausschließlich auf das folgende Dokument:

[SoKl01] F. Soremski, H. Kleinbub: Fahrzeugbedarfsplanung, Hausarbeit im Fach Anwendung der betrieblichen Systemforschung im Wintersemester 2000/2001, Konstanz, 2001.

# Abbildungsverzeichnis

[Abbildung 1 - Fahrzeugbedarfsplanung 4.0 nativ auf Windows 10 – Julian Beck 7](#_Toc440291905)

[Abbildung 2 - Anzeige des Fahrzeugbedarfs – Julian Beck 7](#_Toc440291906)

[Abbildung 3 - der neue Solver "LP Solve" kann ausgewählt werden – Julian Beck 8](#_Toc440291907)

[Abbildung 4 - ohne Optimierung (Abb. Version 2.0, Windows XP) - SoKl01 9](#_Toc440291908)

[Abbildung 5 - mit Optimierung (Abb. Version 2.0, Windows XP) - SoKl01 9](#_Toc440291909)

[Abbildung 6 - Ansatz und Lösung mit Maschinenbelegung - SoKl01 10](file:///C:\Users\Büro\ownCloud\Studium\ALO\Ausarbeitung_Fahrzeugbedarfsplanung_4.0.docx#_Toc440291910)

[Abbildung 7 - LP-Ansatz zur Fahrzeugplanung für 2 und 3 Touren - SoKl01 10](file:///C:\Users\Büro\ownCloud\Studium\ALO\Ausarbeitung_Fahrzeugbedarfsplanung_4.0.docx#_Toc440291911)

[Abbildung 8 - Schritte eines Algorithmus - SoKl01 11](file:///C:\Users\Büro\ownCloud\Studium\ALO\Ausarbeitung_Fahrzeugbedarfsplanung_4.0.docx#_Toc440291912)

[Abbildung 9 - Bedarfsvector - SoKl01 12](file:///C:\Users\Büro\ownCloud\Studium\ALO\Ausarbeitung_Fahrzeugbedarfsplanung_4.0.docx#_Toc440291913)

[Abbildung 10 - Ansatz für Solver als Zähler - SoKl01 13](#_Toc440291914)

[Abbildung 11 - Beispiel für Tourverschiebung - SoKl01 14](file:///C:\Users\Büro\ownCloud\Studium\ALO\Ausarbeitung_Fahrzeugbedarfsplanung_4.0.docx#_Toc440291915)

1. Es ist egal, ob mehrere Touren in einer Periode beginnen. Es kann an Beispielen gezeigt werden, dass der Algorithmus trotzdem zu einer korrekten Lösung kommt. [↑](#footnote-ref-2)
2. Abbildung 7 zeigt, wie die oberste helle Tour, die eindeutig ersichtlich mit Bus 1 fahren wird (Grafik 1), in Grafik 2 und 3 eigentlich von der dunklen Tour gelöscht wird, aber in Grafik 4 doch mit Bus 1 fährt. [↑](#footnote-ref-3)