Bedienungsanleitung XFVRP

Inhalt

[1. Einleitung 1](#_Toc337914501)

[2. Leistungsumfang 2](#_Toc337914502)

[2.1 Touren- oder Fahrzeugplanung 2](#_Toc337914503)

[2.2 Heterogener Fuhrpark 2](#_Toc337914504)

[3. Für Anwender: Aufruf des XFVRP-Programms 5](#_Toc337914505)

[3.1 Vorbereitung der Daten in DISMOD 5](#_Toc337914506)

[3.2 Vorbereitung der Daten in Excel 5](#_Toc337914507)

[3.2.1 Aufruf 9](#_Toc337914508)

[3.2.2 Ergebnisse 9](#_Toc337914509)

[3.3 Prämissen 10](#_Toc337914510)

[4. Verwendung der XFVRP-Bibliothek (für Entwickler) 11](#_Toc337914511)

[4.1 Flexi-Importer 11](#_Toc337914512)

[4.2 Relevante Klassen und Methoden 12](#_Toc337914513)

[4.3 Beispiele 14](#_Toc337914514)

[4.4 Metriken 15](#_Toc337914515)

[4.5 Report 16](#_Toc337914516)

[4.6 Optimierungsverfahren 16](#_Toc337914517)

[Konstruktionsverfahren 17](#_Toc337914518)

[Verbesserungsverfahren 18](#_Toc337914519)

# Einleitung

Tourenplanung gehört zu den Standardwerkzeugen der Optimierung von Transportstrukturen. Es wird für die Nahverkehrsdistributionsplanung genauso eingesetzt wie für die Fernverkehrsplanung in Stückgutnetzwerken. Auch eher abgelegene Themen wie intra-logistische Fördersysteme oder der öffentliche Nahverkehr beruhen auf der Fragestellung, wie ein günstiger Transport und die Rückführung von Material und Ressourcen zu bewältigen sind.

XFVRP (eXtremly Fast Vehicle Routing Planning) ist eine Tourenplanung aus der Abteilung Verkehrslogistik, die speziell für Bearbeitung taktischer oder strategischer Fragestellungen konzipiert wurde. Im Fokus stehen die Einfachheit der Benutzung und im Verständnis der zugrunde liegenden Algorithmen. Der Umfang der damit lösbaren Tourenplanungsprobleme ist klar definiert und umfasst die gängigsten Restriktionen und Erweiterungen.

Die beispielhafte Benutzeroberfläche ist durch die Verbindung mit Excel schnell für die Projektbearbeiter zugänglich und führt aufgrund der implementierten Verfahren schnell zu guten bis sehr guten Ergebnissen. Die XFVRP-Suite ist von seiner Konzeption als Modul gut in bestehende Planungsplattformen integrierbar. Der Aufbau als Framework mit ausführlicher Dokumentation ermöglicht den schnellen Zugang zu den Planungsmethoden.

Die Software und der für die Abteilung frei zugängliche Quellcode des XFVRP-Frameworks liegt im SVN-Repository „svn+ssh://leitstand.iml.fhg.de/proj/vproject.32x/Aktuelle/0001-DB-Server/svn320/DEV320/Prototypes/branches/XFVRP8„ .

# Leistungsumfang

## 2.1 Touren- oder Fahrzeugplanung

Eine Tourenplanung bestimmt für eine Menge von Sendungen, die optimale Zuordnung zu Touren ohne die gesetzten Restriktionen zu verletzen. Die Anzahl der Touren wird im Allgemeinen mit der Anzahl der Fahrzeuge gleichgesetzt.

Das XFVRP-Paket kann folgende Aspekte einer Tourenplanung berücksichtigen:

* Ein-Depot- und Mehr-Depot-Planung
* Geschlossene und Offene Touren
* Gemischte Be- und Entladung auf einer Tour (beschränktes Pickup & Delivery)
* Heterogener Fuhrpark (Heuristischer Ansatz!) [siehe Abschnitt 2.2]
* Drei Mengenangaben pro Sendung
* Drei obere Kapazitätsschranke pro Fahrzeugtyp
* Maximale Tourendauer
* Maximale Anzahl Stopps pro Tour
* Zwei Zeitfenster pro Auftrag/Kunde oder Depot
* Explizite Angabe von Umschlagszeiten je Sendung
* Laderaumrestriktionen (siehe Abschnitt Laderaumplanung)
* Entfernungen zwischen den Standorten
  + vorberechnete Entfernungsmatrizen
  + über Koordinaten mit Projektion auf Erdkugel
  + über Koordinaten euklidisch nach Pythagoras
* Stichtouren für á priori ungültige Sendungen
* Fixkosten für Fahrzeuge zur Reduktion der Tourenanzahl
* Vorgabe einer Anfangslösung (siehe Abschnitt XX)
* Vorgabe von strukturellen Restriktionen (siehe Abschnitt XX)
* Zielfunktion: Minimierung der Gesamtfahrstrecke unter Bewahrung aller Restriktionen

## 2.2 Heterogener Fuhrpark

Bei der Planung mit verschiedenen Fahrzeugtypen wird eine vereinfachte Planungsheuristik verwendet.

Folgende Annahmen sind bei einer solchen Planung zu beachten:

* Beschränkter Fuhrpark  
  Es wird davon ausgegangen, dass pro Fahrzeugtyp nur endlich viele Fahrzeuge zur Verfügung stehen. Bei Sub-Dienstleistern muss dies in der Praxis nicht zwangsläufig sein.
* Priorisierung der Fahrzeugtypen  
  Es wird davon ausgegangen, dass vor der Planung klar festgelegt werden kann, welcher Fahrzeugtyp klar gegenüber einem anderen Fahrzeugtyp zu bevorzugen ist.
* Gleiche Fahrzeiten  
  Der aktuelle Entwicklungsstand sieht für alle Fahrzeugtypen die gleiche Metrik vor, so dass alle Fahrzeuge gleich schnell vom Verfahren bewertet wird.

Das Verfahren arbeitet folgendermaßen:  
Die Menge der Fahrzeugtypen wird in eine zu bearbeitende Reihenfolge gebracht (das Fahrzeug mit dem größten Laderaum (Priorisierungskriterium), dann das mit dem zweitgrößten und so weiter). Pro Fahrzeugtyp in der Reihenfolge wird für die aktuelle Menge an aktiven Sendungen mit den Parametern des Fahrzeugtyps eine reguläre Tourenplanung durchgeführt. Anschließend werden die resultierenden Touren anhand ihrer Auslastung und Länge in eine Reihenfolge gebracht (absteigend nach Auslastung)und die ersten k besten Touren gewählt, wobei k die Anzahl der zur Verfügung stehenden Fahrzeuge für diesen Fahrzeugtyp ist. Die Priorisierung dieser Touren findet anhand des Quotienten aus Auslastung und Tourenkosten statt. Alle Sendungen auf den nicht gewählten Touren werden in die nächste Untersuchung mit dem nächsten Fahrzeugtyp verschoben. Bleiben nach Bearbeitung aller Fahrzeugtypen noch Aufträge übrig, werden für diese Stichtouren (mit Kennzeichen INVALID) erzeugt.

**2.3 Laderaumplanung**

Die Laderaumplanung entspricht den im Forschungsprojekt Efficient Load entwickelten Planungsalgorithmen. Hierbei werden aus den Touren der Tourenplanung die Sendungen ermittelt und in Packstücke und Laderäume (entspr. des Fahrzeugtyps) extrahiert. Für die Packstücke wird eine grundlegende Annahme getroffen, dass Packstücke in der Realität Ladungsträger abbilden, die nicht weiter zu Gebinden zusammengestellt werden müssen. Die hier vorliegende Laderaumplanung beherrscht keine Palettierung von Kleinstpackstücken.

Während der Tourenplanung wird eine integrierte Laderaumplanung pro Tour durchgeführt, um neben den gewichts- und volumenmäßigen Kapazitätsrestriktionen zusätzliche Faktoren der Laderaumplanung abzuprüfen. Zur Beschleunigung der Planung wird die Laderaumplanung nur dann aufgerufen, wenn eine Neubewertung wirklich notwendig ist.

Ein wichtiger Aspekt der Laderaumplanung ist die Stapelung. Da die Packstücke Ladungsträger repräsentieren, wird eine generelle Stapelfähigkeit unterstellt. Im Ergebnis werden neben den Packstücken auch die gebildeten Stapel ausgegeben.

Die Laderaumplanung berücksichtigt folgende Eigenschaften:

* Eine Sendung kann aus mehreren Packstücken bestehen.
* Jedes Packstück wird über drei Dimensionen in seinen Ausmaßen definiert: Länge, Höhe, Breite.
* Jeder Laderaum wird über drei Dimensionen in seinen Ausmaßen definiert: Länge, Höhe, Breite. Zudem besitzt eine Laderaum nur eine einzige Ladeseite, über den der Laderaum zugänglich ist.
* Jedes Packstück hat ein Eigengewicht und eine maximale Belastbarkeit.
* Jedes Packstück muss be- und entladen werden. (Auch am Depot)
* Über die Tourenplanung können die Packstücke in eine Be- und Entladereihenfolge gebracht werden.
* Jedem Fahrzeug einer Tour stehen bis zu zwei Laderäume zur Verfügung. Dies ist abhängig vom aktiven Fahrzeugtyp.

Folgende Restriktionen werden von der Laderaumplanung berücksichtigt:

* Packstücke im Laderaum dürfen sich gegenseitig nicht durchdringen.
* Packstücke dürfen nicht über die Außenwände des Laderaums ragen.
* Die Summe der Gewichte der gepackten Packstücke in einem Laderaum darf die maximale Zuladung zu keinem Zeitpunkt überschreiten.
* Die Belastung eines Packstückes durch darauf gepackte Packstücke darf zu keinem Zeitpunkt überschritten werden.
* Wenn die Zugänglichkeitsrestriktion aktiv ist, muss beim Entladen das Packstück direkt von der Ladeseite des Laderaums zugänglich sein, ohne das andere Packstücke den Weg versperren.

Die Laderaumplanung ist standardmäßig nicht aktiviert. Sie wird durch eine Parametrisierung der Eingabe aktiviert.

**2.4 Vorgabe struktureller Restriktionen**

Für folgende beispielhafte Aufgabenstellungen kann das XFVRP strukturelle Restriktionen bei der Optimierung mitberücksichtigen:

* Zur Validierung und Plausibilisierung von XFVRP-Lösungen
* Zur partiellen Optimierung von bestehenden Tourenplänen (dynamische Optimierung in operativen Einsätzen)
* Bei Restriktionen, die bislang noch nicht vom XFVRP berücksichtigt werden können.

Im Weiteren sollen die verschiedenen Arten der strukturellen Restriktionen beschrieben werden:

* Vorgabe von Tourenplänen als Startlösung:  
  Dem XFVRP kann in Form einer formatierten Zeichenkette eine teilweise oder vollständige Lösung einer Eingabe mitgegeben werden. Diese vorgegebene Lösung wird beim Aufbau der ersten Lösung im XFVRP erzeugt und bewertet. Ist die Lösung nur teilweise, also nicht alle Aufträge der Eingabe werden festgelegt, werden die nicht festgelegten Aufträge als Stichtouren aufgebaut. Besonders im Zusammenhang mit einer Parametrisierung, bei der keine Optimierung durchgeführt werden soll, kann so vom Benutzer eine manuelle Prüfung unerwarteter Ergebnisse des XFVRP vorgenommen werden.
* Vorgabe von Fahrzeugtypen  
  Einem Auftrag können ein oder mehrere Fahrzeugtypen über eine formatierte Zeichenkette zugewiesen werden. Das XFVRP zwingt den Auftrag von diesem Fahrzeugtyp verwendet zu werden. Wird einem Auftrag kein Wert zugeordnet, so können alle Fahrzeugtypen von der Planung verwendet werden.
* Vorgabe von Tourenblöcken  
  Ein Auftrag kann ein Blockname mit übergeben werden. Alle Aufträge mit dem selben Blocknamen müssen von der XFVRP-Planung auf einer einzigen Tour sein. Die Touren können dabei sowohl aus geblockten und ungeblockten Aufträgen bestehen. Aufträge ohne zugewiesenen Blocknamen sind nicht eingeschränkt.
* Vorgabe eines Auftragsranges  
  Einem Auftrag kann ein Rang (oder auch Priorität) zugewiesen werden, der eine generelle Reihenfolge innerhalb eines Tourblockes ausdrückt. Ein Auftrag mit einem höheren Rang darf in der Reihenfolge einer Tour nicht vor einem Auftrag mit einem niedrigeren Rang liegen. Zwischen zwei priorisierten Aufträgen können sich jedoch weitere unpriorisierte Aufträge befinden. Aufträge ohne Rang sind frei in der Platzierung in den Touren. Hat ein Auftrag mit Rang gleichzeitig keinen zugewiesenen Blocknamen, gilt das Kriterium zwar weiterhin, aber priorisierte Aufträge können auf unterschiedlichen Touren liegen. Damit wird die Restriktion abgeschwächt.
* Vorgabe einer Auftragssequenzposition  
  Diese Restriktion ist eine Verschärfung des Auftragsranges. Hierbei wird eine feste Reihenfolge der Aufträge im Tourenplan festgelegt. Zwei Aufträge mit direkt aufeinander folgenden Sequenzpositionen müssen im Tourenplan aufeinander innerhalb einer Tour folgen. Die gilt auch für Aufträge ohne Blockung. Unsequenzierte Aufträge sind frei von dieser Restriktion und können in beliebiger Reihenfolge verplant werden.

# Für Anwender: Aufruf des XFVRP-Programms

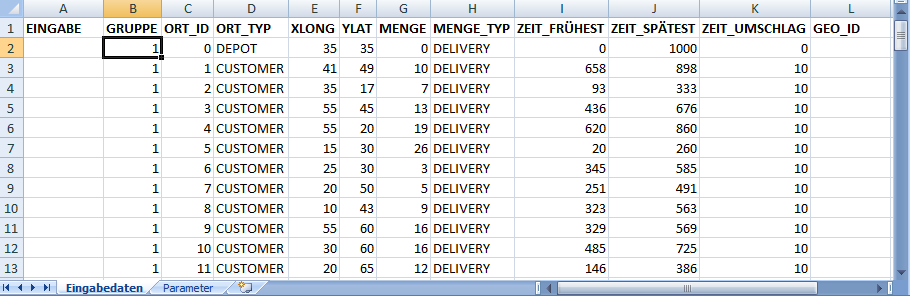
## 3.1 Vorbereitung der Daten in DISMOD

Die Schnittstelle zu DISMOD wurde aufgrund des hohen Alters und der eingeschränkten Pflege aufgegeben. Hier gilt es von Seiten der Plattformbetreiber eine geeignete Schnittstelle zu entwerfen.

## 3.2 Vorbereitung der Daten in Excel

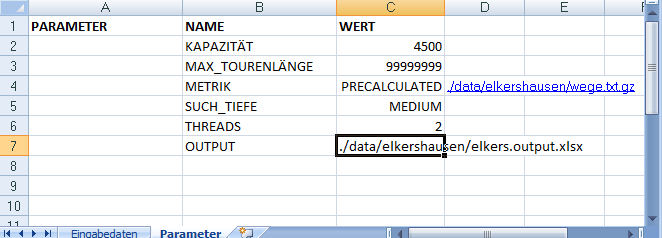
Grundlage der Berechnungen ist das Excel-Dokument „xfvrp.xls“, welches den Programmdaten beiliegt. Es enthält drei Reiter: „Eingabedaten“, „Parameter“ und „Flotte“. Unter Eingabedaten befindet sich eine exemplarisch gefüllte Tabelle mit einer Überschriftenzeile und den Auftragsdaten. Die Reihenfolge der Spalten darf verändert werden. Die Auftragsdaten können vom Benutzer verändert werden.

* Durch die Werte in der Spalte „GRUPPE“ können mehrere Tourenplanungsaufrufe über eine Dateneingabe aufgerufen werden. Alle Sendungen einer Gruppe werden zusammen in einer Tourenplanung berücksichtigt. Je Gruppe muss es einen Depot-Standort geben. Sind mehrere angegeben, so wird eine Mehr-Depot-Standortplanung durchgeführt, die mehr Planungsaufwand benötigt.
* Die „ORT\_ID“ ist eine für alle Zeilen eindeutig zu vergebene Nummer, die die einzelne Zeile später im Ergebnis repräsentiert. Es handelt sich hierbei um einen frei verwendbaren Wert, der zum Beispiel die Nummer des Kunden oder aber auch eine Auftragsnummer beschreibt.
* „ORT\_TYP“ bezeichnet die Art des Ortes oder Auftrags, wobei es nur zwei mögliche Belegungen geben darf: „DEPOT“ bezeichnet den Standort des Depot-Standortes und „CUSTOMER“ steht für Orte, an denen Menge Be- oder Entladen wird. Bei der Angabe von Sendungsaufträgen muss in dieser Spalte der Wert CUSTOMER verwendet werden.
* „XLONG“ ist die Longitude aus einem digitalen Kartenwerk z.B. google maps / Format: WGS84
* „YLAT“ ist die Latitude aus einem digitalen Kartenwerk z.B. google maps / Format WGS84
* „MENGE“ oder „MENGE\_1“ gibt die zu lieferende oder abzuholende Menge am Kundenstandort an. Am Depot selbst wird dieser Wert ignoriert. Die erste Mengenkapazität die einem Fahrzeugtyp zugewiesen wird, bezieht sich auf diesen Mengenwert. Dieser Wert ist ein beliebiger Zahlenwert.
* „MENGE\_2“ ist ein weiterer Mengenwert und korreliert mit der zweiten Mengenkapazität eines Fahrzeugtyps. Dies ist ein beliebiger Zahlenwert.
* „MENGE\_3“ ist ein weiterer Mengenwert und korreliert mit der dritten Mengenkapazität eines Fahrzeugtyps. Dies ist ein beliebiger Zahlenwert.
* „MENGE\_TYP“ gibt an, ob es sich bei dem Besuch am CUSTOMER-Ort um eine Belieferung(„DELIVERY“) oder Abholung(„PICKUP“) handelt.
* „ZEIT\_FRÜHEST“ gibt einen ganzzahligen Wert für die früheste Öffnungszeit/Bereitstellungszeit am CUSTOMER-Ort an.
* „ZEIT\_SPÄTEST“ gibt einen ganzzahligen Wert für die späteste Schlusszeit/Bereitstellungszeit am CUSTOMER-Ort an.
* „ZEIT\_UMSCHLAG“ gibt einen ganzzahligen Wert, der für den Umschlag dieses Auftrages notwendig ist.
* „GEO\_ID“ wird bei vorberechneten Entfernungswerken benötigt. In diesem Fall können Sendungen bestimmte geografische Orte als numerischen Wert zu geordnet werden.
* Bei „ANZAHL\_PAKETE“ wird für eine Sendung die Anzahl der Einzelpakete angegeben. Wird nur berücksichtigt, wenn Laderaumplanung aktiv ist.
* „PAKET\_GEWICHT“ gibt das Gewicht eines Paketes an. Wird nur berücksichtigt, wenn Laderaumplanung aktiv ist.
* „PAKET\_LÄNGE“ gibt die Länge eines Paketes an. Wird nur berücksichtigt, wenn Laderaumplanung aktiv ist.
* „PAKET\_BREITE“ gibt die Breite eines Paketes an. Wird nur berücksichtigt, wenn Laderaumplanung aktiv ist.
* „PAKET\_HÖHE“ gibt die Höhe eines Paketes an. Wird nur berücksichtigt, wenn Laderaumplanung aktiv ist.
* „PAKET\_AUFLAST“ gibt die akzeptierte Belastbarkeit eines Paketes an. Wird nur berücksichtigt, wenn Laderaumplanung aktiv ist.
* Weitere Spalten sind „VORGABE\_FAHRZEUG“, „VORGABE\_BLOCK“, „VORGABE\_SEQUENZ\_RANG“ sowie „VORGABE\_SEQUENZ\_POS“ die im Kapitel 2.4 näher erläutert werden.



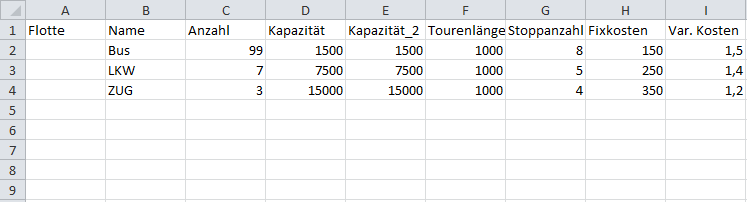
Im zweiten Reiter befinden sich die Angaben für die Planungsparameter, die vom Anwender eingeben werden müssen. Für viele Parameter gibt es default-Werte, die ohne Bedenken so stehen gelassen werden können. Zu den Parametern gehören:

* „KAPAZITÄT“ gibt die maximale Kapazität des Fahrzeugs an. Dieser Wert richtet sich nach den Mengenangaben im Eingabedaten-Reiter. Je nachdem ob Kilogramm, Tonnen oder Kubikmeter geplant werden, ist hier ein anderer Wert einzutragen. Bei Angabe einer Fahrzeugflotte durch einen separaten Reiter wird dieser Wert durch die fahrzeugtyp-spezifischen Parameter überschrieben.
* „MAX\_TOURENDAUER“ beschreibt die maximale Dauer einer Tour. Hierbei wird der Fahrzeitwert inklusive der Umschlagszeiten und Wartezeiten beschränkt. Bei Angabe einer Fahrzeugflotte wird dieser Wert durch die fahrzeugtyp-spezifischen Parameter überschrieben.
* Über den Parameter „METRIK“ lässt sich einstellen, ob mit dem Wert „AIR“ die Luftlinienentfernung zwischen zwei Knoten mit angegebenen Geokoordinaten berechnet wird oder ob die euklidische Entfernung (Pythagoras) per „EUCLIDEAN“ eingesetzt wird. Mit dem Wert „PRECALCULATED“ kann in einer weiteren Spalte der Pfad zu einer Text-Datei angegeben werden, in der für jede Kombination von vorhandenen Geoids aus dem Reiter „Eingabedaten“ ein Entfernungs- und Fahrzeitwert hinterlegt ist.
* Die „SUCH\_TIEFE“ bestimmt, wie stark die Optimierung die Potenziale versucht auszuschöpfen. Mit „LOW“ werden die gröbsten Optimierungsschritte auch für große Problemstellungen schnell durchgeführt. Mit „MEDIUM“ wird bereits ein gutes bis sehr gutes Ergebnis erreicht, wobei die Laufzeit durch ansteigt. Auf der Stufe „HIGH“ werden alle Optimierungsverfahren, die dem XFVRP-Paket zur Verfügung stehen, eingesetzt, was jedoch gerade bei größeren Instanzen zu unakzeptablen Laufzeiten führen kann.
* Mit „THREADS“ lässt sich die Anzahl der zur Planung verwendeten Prozessoren angeben. Auf einem Rechner mit zwei Prozessoren (kann im Task-Manager überprüft werden) sollte am besten ein Wert zwei eingetragen werden. Dieser Wert darf nicht kleiner 1 sein und wenn mehr Threads eingesetzt werden sollen als Prozessoren zur Verfügung stehen, so wird die Optimierung durch den nötigen Overhead eher verlangsamt.
* Im Parameter „OUTPUT“ wird der Pfad und Dateiname der Ausgabedatei angegeben.
* Allein durch die Angabe dieses Parameters „MIT\_BELADUNG“ wird die Laderaumplanung aktiviert. Werden im Eingabedaten-Blatt keine expliziten Angaben zu den Packstücken gemacht, werden Standardwerte genommen. Dieser Parameter erwartet keinen zusätzlichen Wert.
* Der Parameter „OHNE\_REIHENFOLGE\_BEI\_DER\_BELADUNG“ deaktiviert die sonst standardmäßige Berücksichtigung der Entladereihenfolge bei der Beladung, um so mögliche Rangiertätigkeiten während der Tourdurchführung zu reduzieren.
* Durch die Parameter „TOUREN\_START\_OFFEN“ und „TOUREN\_ENDE\_OFFEN“ können Touren mit offenen Start und/oder Ende bei der Planung eingestellt werden. Diese Parameter haben keinen zugehörigen Wert. Werden beide Werte angegeben, so haben die Touren kein Start oder Ende.



Wenn der Reiter „Flotte“ angegeben wird, arbeitet die Tourenplanung mit einer heterogenen Flotte. Hierbei ist zu beachten, dass von einer beschränkten Flotte ausgegangen wird, bei der eine Hierarchie der Priorisierung der Fahrzeugtypen vorliegt, so dass günstige Fahrzeugtypen vollständig versucht werden, einzusetzen. Hierfür müssen die Parameter je Fahrzeugtyp angegeben werden:

* „NAME“ enthält den Namen des Fahrzeugtyps, der in den Ergebnissen am Abschluss der Planung ausgegeben wird.
* „ANZAHL\_FAHRZEUGE“ gibt die vorhandene Menge an Fahrzeugen dieses Typs an, die von der Tourenplanung verwendet werden können.
* „KAPAZITÄT“ oder „KAPAZITÄT\_1“ beziffert die Zuladungskapazität des Fahrzeugtyps.
* „KAPAZITÄT\_2“ beziffert die Zuladungskapazität des Fahrzeugtyps für die zweite Mengenangabe.
* „KAPAZITÄT\_3“ beziffert die Zuladungskapazität des Fahrzeugtyps für die dritte Mengenangabe.
* „ANZAHL\_STOPS\_PRO\_TOUR“ gibt die maximale Dauer einer Tour an.
* „ANZAHL\_STOPS\_PRO\_TOUR“ gibt die maximale Anzahl von unterschiedlichen Lade- bzw. Entladestellen auf einer Tour an.
* Die „FIXKOSTEN“ werden pro nicht-leere Tour des betreffenden Fahrzeugtyps angesetzt.
* Die „VARKOSTEN“ werden dagegen pro gefahrenen Kilometer Strecke für eine Kostenbewertung angesetzt.
* „HÖHE\_LADERAUM\_1“ und „HÖHE\_LADERAUM\_2“ geben die Höhe für den ersten oder zweiten Laderaum eines Fahrzeugs an.
* „BREITE\_LADERAUM\_1“ und „BREITE\_LADERAUM\_2“ geben die Breite für den ersten oder zweiten Laderaum eines Fahrzeugs an.
* „LÄNGE\_LADERAUM\_1“ und „LÄNGE\_LADERAUM\_2“ geben die Länge für den ersten oder zweiten Laderaum eines Fahrzeugs an.



### 3.2.1 Aufruf

Nachdem die Excel-Tabelle ausgefüllt wurde, muss diese geschlossen werden, damit das externe Planungsprogramm darauf zu greifen kann. Die Excel-Tabelle muss ebenfalls im selben Ordner wie das auszuführende Programm (also bei start.vrp.bat) liegen.

Nun kann das Programm „start.vrp.bat“ durch einen Doppelklick gestartet werden. Es öffnet sich ein Konsolenfenster in dem die aktuellen Berechnungsfortschritte dargestellt werden. Schließt sich das Fenster und es sind keine Fehler während der Berechnung aufgetreten, wurde die Ergebnisdatei an die in den Parametern angegebene Stelle im Dateisystem geschrieben.

### 3.2.2 Ergebnisse

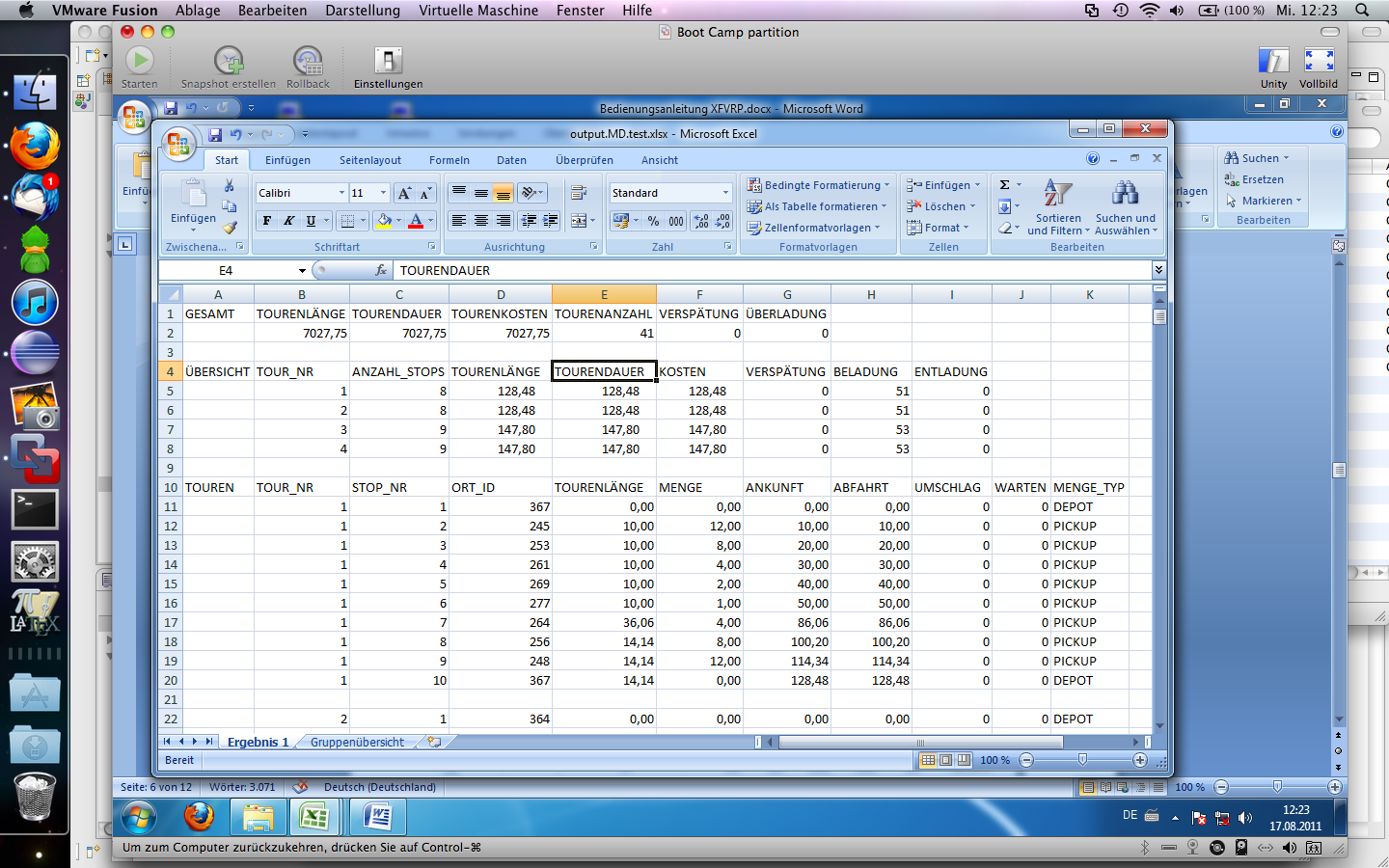
Für jeden Planungslauf wird eine (oder mehrere zusammenhängende) Ergebnisdatei/en geschrieben. Für jede Planungsgruppe (siehe Spalte „GRUPPE“ in der Eingabe) wird ein Ergebnis-Blatt erstellt. In dem Reiter „Gruppenübersicht“ werden die aggregierten Gesamtergebnisse der einzelnen Gruppen aufgelistet.

Die Ergebnisdarstellung einer Gruppe im Reiter „Ergebnis“ gliedert sich in drei Blöcke. Zunächst werden im obersten Block die Kennwerte des Tourenplans mit der Gesamttourenlänge, der Anzahl der benötigten Touren, der Summe der Verspätungen, wenn Z eitfenster angegeben wurde sowie der unvermeidbaren Überladung in den Spalten aufgeführt.

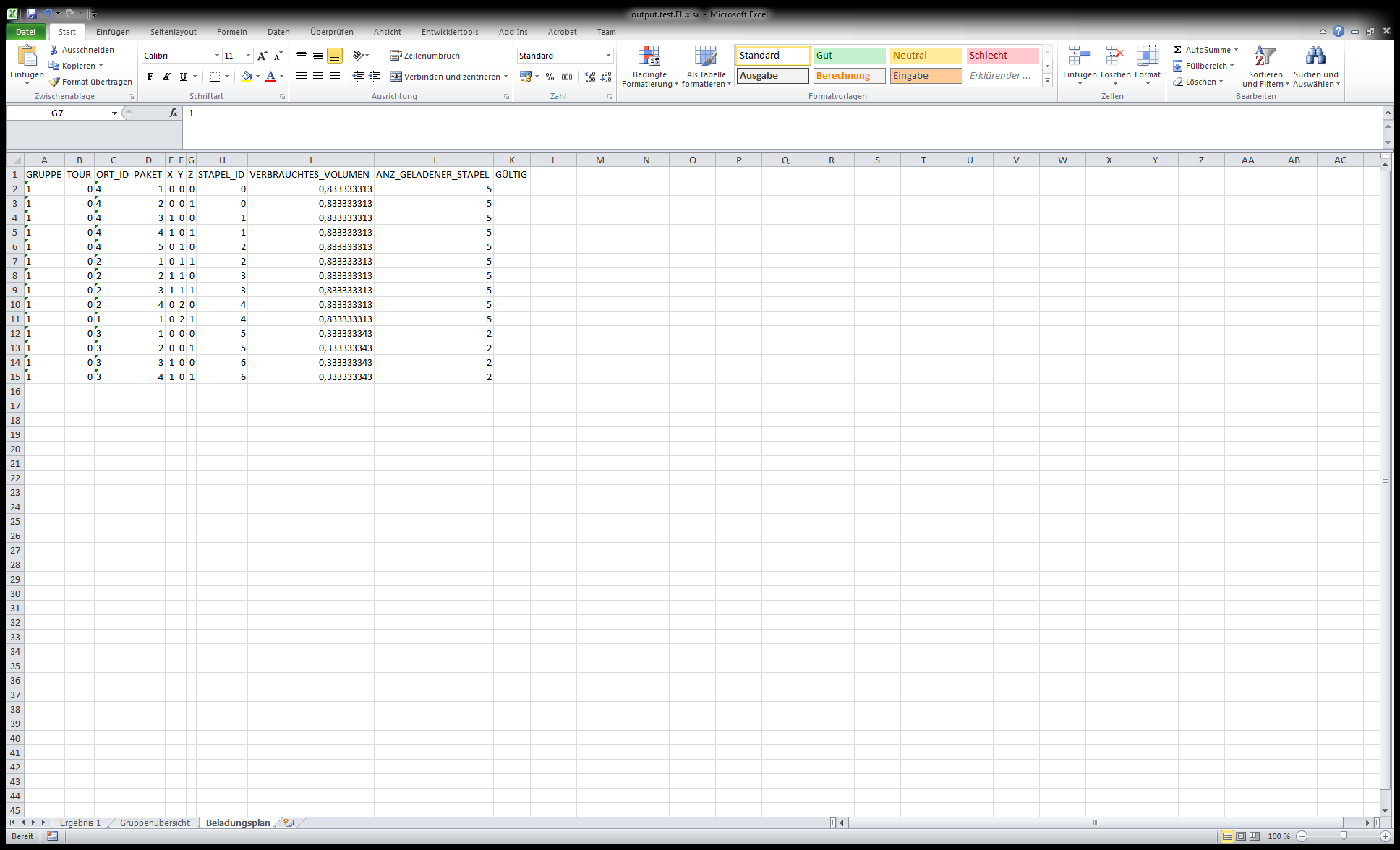
Darunter werden im zweiten Block die Kennwerte pro Tour mit der Anzahl der Stops, der jeweiligen Tourenlänge, der Verspätung sowie den summierten Mengenwerten für Be- und Entladung angegeben.

Anschließend folgen im dritten Block die Einzelauswertungen der Touren. Dabei wird pro Tour die Sequenz der Kundenbesuche aufgelistet, wobei pro Zeile die Tournummer, die Stopnummer, die ID des Kundens/Auftrags, die seit dem letzten Ort zurückgelegten Entfernung, die be- oder entladene Menge, die Ankunftszeit in Minuten, die Abfahrtszeit in Minuten, die Umschlagszeit in Minuten, die Wartezeit, falls der Auftrag bei Ankunft noch nicht zur Verfügung stand, sowie einem Flag, um welche Art von Ereignis es sich in dieser Zeile handelt. Bei „DEPOT“ wurde in dieser Zeile der Besuch an einem Depot beschrieben. Bei „PICKUP“ wurde etwas beladen und bei „DELIVERY“ wurde etwas entladen.

Je nachdem ob die Eingabedaten mehrere Mengenangaben enthielten, werden auch nur die verwendeten Mengenangaben im Report ausgegeben.

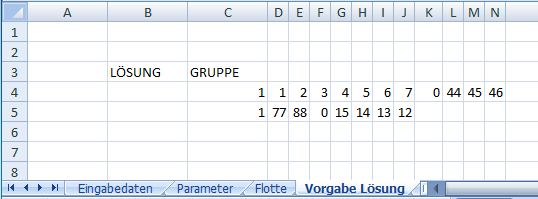


Wenn die Tourenplanung mit der Laderaumplanung durchgeführt wurde, wird im Ergebnis auch der resultierende Beladungsplan im Reiter „Beladungsplan“ angegeben. Für jede Gruppe und für jede Tour wird detailliert die Beladungssequenz der Sendungen angegeben. Für jede Sendung werden dessen Packstücke, mit den Raumkoordinaten an denen sie im Laderaum gesetzt wurden, ausgegeben. Da Packstücke in Stapeln verplant werden, wird jedem Packstück die ID eines Stapels zugewiesen. Wenn Packstücke aufgrund der Restriktionen nicht geladen werden konnten, werden sie in der Spalte „GÜLTIG“ als „UNGÜLTIG“ markiert und erhalten keine Raumkoordinaten.



### 3.2.2 Vorgaben von Startlösungen

Um dem XFVRP eine Lösung vorzugeben, kann vom Benutzer ein weiterer Reiter angelegt werden. Dieser Reiter muss den Namen „Vorgabe Lösung“ haben, damit er von der Planung berücksichtigt wird. Die Lösung besteht aus einer zeilenweise angegebenen Tourenbeschreibung. Zunächst muss pro Zeile die Gruppe angegeben werden, für die eine Lösung beschrieben wird. Anschließend folgt in direkter Reihenfolge die ORT\_IDs der Aufträge, die zu dieser Tour gehören. Nicht alle Aufträge einer Gruppe müssen dabei aufgeführt sein. Die folgende Darstellung zeigt den Aufbau einer vorgegebenen Lösung.



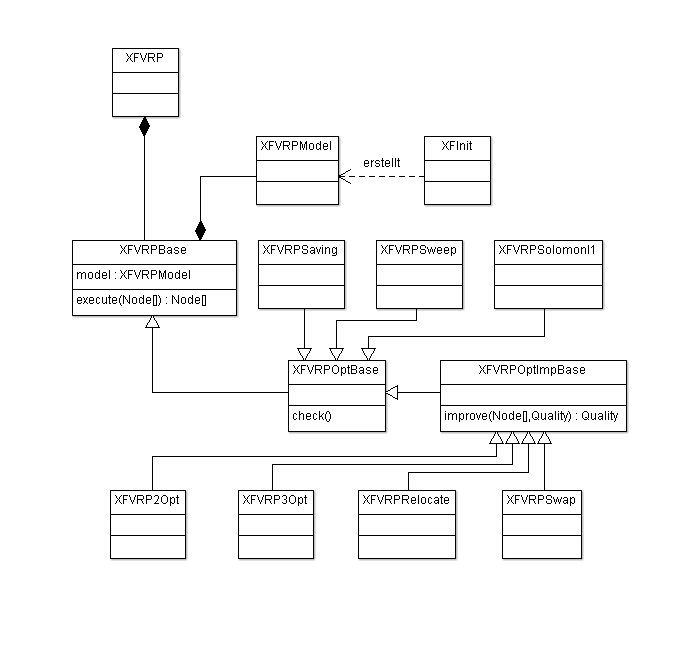
## 3.3 Prämissen

XFVRP wurde in der Programmiersprache JAVA erstellt und benötigt für die Ausführung dieses Programms eine Version höher als 1.5 . Eine aktuelle Version kann unter <http://java.sun.com/javase/downloads/index.jsp> aus dem Internet geladen.

# Verwendung der XFVRP-Bibliothek (für Entwickler)

Es gibt zwei Stufen der Quellcode-Verwendung. Um den Quellcode in eine Umgebung/Plattform einzubauen, reicht es aus, sich die Klasse XFVRP anzuschauen, die im Folgenden ausführlich beschrieben wird.

Um eigene Verfahren oder Varianten der Optimierungsmethoden zu entwickeln, gilt es tiefer einzusteigen. Hierfür ist die Klasse XFVRPBase und deren Subklassen interessant, die hier aber nicht erläutert sind. Es sei auf das Javadoc im „doc“-Verzeichnis verwiesen.



## 4.1 Flexi-Importer

Mit der Integration der Laderaumplanung wurde es notwendig eine hohe Anzahl von Parametern flexibel in das XFVRP-System einzufügen. Hierfür wurde das Flexi-Importer-System entwickelt, welches es dem Anwender möglich macht, Parameter zu Depots, Kunden und Fahrzeugen in beliebiger Reihenfolge und beliebiger Anzahl anzugeben.

Der Flex-Importer ist ein der Tourenplanung vorgelagertes System, bei dem zunächst die Daten des Anwenders gesammelt werden. Erst beim Aufruf der Tourenplanung werden die Daten vom FlexiImporter-Objekt in der Klasse XFVRP am Block abgerufen. Der Flexi-Importer organisiert zunächst also die Pufferung der Daten in einer flexiblen Meta-Datenstruktur, bevor sie für die Tourenplanung in feste Datenstrukturen überführt werden.

Die zweite Eigenschaft des Flexi-Importers ist die Art wie Daten dem XFVRP-System bekannt gemacht werden. Über die Methoden XFVRP .addDepot(), XFVRP .addCustomer() und XFVRP .addVehicle() werden Datenobjekte vom FlexiImporter im XFVRP-Objekt aufgerufen, die eine Datenschnittstelle zum Eingeben neuer Daten darstellen. Über selbstreflektierende Set-Methoden können dem Daten-Objekt Daten eingegeben werden:

vrp

.addDepot()

.setMethode1(wert1)

.setMethode2(wert2)

.setMethode1(wert2);

Die Wiederholung einer Set-Methode ist prinzipiell möglich, wobei der zweite Wert den ersten überschreibt. Sinnvoll ist das jedoch nicht.

## Relevante Klassen und Methoden

Eine ausführliche Beschreibung der zur Verfügung stehenden Methoden in der XFVRP-Schnittstelle können in der beigefügten API-Dokumentation eingesehen werden.

Im Folgenden werden die wichtigsten Methoden zum Betrieb einer XFVRP-Planung beschrieben:

* XFVRP . executeRoutePlanning()  
  Startet die Tourenplanung mit den eingestellten Parametern.
* XFVRP .addDepot()  
  Gibt das Datenobjekt des FlexiImporters zurück, um damit Angaben über ein weiteres Depot zu machen.
* XFVRP .addCustomer()  
  Gibt das Datenobjekt des FlexiImporters zurück, um damit Angaben über einen weiteren Kundenauftrag(Sendung) zu machen.
* XFVRP .addVehicle()  
  Gibt das Datenobjekt des FlexiImporters zurück, um damit Angaben über einen weiteren Fahrzeugtyp zu machen. Die Parameter für die homogene Planung setCapacity() und setMaxTourLength() werden dabei überschrieben.
* XFVRP. clearDepots()  
  Löscht alle bisher angegebenen Depots aus dem System.
* XFVRP. clearCustomers ()  
  Löscht alle bisher angegebenen Kunden/Aufträge aus dem System.
* XFVRP. clearVehicles()  
  Löscht alle bisher angelegten Fahrzeuge.
* XFVRP. setMetric (Metric m)  
  Die Berechnung der Entfernungen wird nun von einem extern steuerbaren Metric-Objekt vorgenommen, welches in die Planung über diese Methode eingefügt werden kann. Ohne Metric-Angabe startet die Planung nicht. Für die Berechnung mit AirMetric oder TriangleMetric (Pytagoras) ist die Angabe von Koordinaten bei den Depots und Customers notwendig.
* XFVRP. setCapactiy (float f) [Default: MAX\_VALUE]  
  Setzt die maximale Kapazität eines Fahrzeugs. Der Wert muss größer als 0 sein. Dabei werden vorher hinzugefügte Fahrzeugtypen gelöscht.
* XFVRP. setMaxRouteDuration(float f)[Default: MAX\_VALUE]  
  Setzt die maximale Tourdauer eines Fahrzeugs. Der Wert muss größer als 0 sein. Dabei werden vorher hinzugefügte Fahrzeugtypen gelöscht.
* XFVRP. setMaxNumberOfStopsPerRoute(int f)[Default: MAX\_VALUE]  
  Setzt die maximal erlaubte Anzahl an Aufträgen(Auftrag gleich Stop) pro Tour. Der Wert muss größer als 0 sein. Dabei werden vorher hinzugefügte Fahrzeugtypen gelöscht.
* XFVRP. getReport ()  
  Gibt einen detaillierten Report des zuletzt berechneten Planungsergebnisses aus (siehe Abschnitt über „Report“). Der Rückgabewert ist ein leerer Report, wenn noch kein fertiges Ergebnis vorliegt.
* XFVRP. allowsAcceleratedMetric ()  
  Zur Beschleunigung der Optimierung kann die Metric des Benutzers in ein effizienteres Format konvertiert werden, so dass die Entfernungsabfrage deutlich fixer läuft. Allerdings wird dadurch deutlich mehr Arbeitsspeicher verbraucht, weswegen Problemstellungen mit sehr vielen Standorten (> 2000 Orte) nicht mit diesem Schalter berechnet werden sollten.
* XFVRP. allowsRoutePlanSplitting()  
  Bei sehr großen Eingabeinstanzen mit mehr als 1000 Kunden ist es sinnvoll um in angemessener Rechenzeit gute Ergebnisse zu erhalten, die Aufgabenstellung zu dekomponieren. Hier wird die Menge an Touren in einem Routenplan in Blöcke zergliedert. Anschließend wird für jeden Block ein eigenes Tourenplanungsproblem gelöst und später die Ergebnisse der Blöcke wieder zu einem ganzen Tourenplan zusammen gefügt. Die durch die Fragmentierung entstehenden Verluste an Lösungsqualität liegen bei unter 1%, wobei im Gegensatz die Laufzeit deutlich absinkt.
* XFVRP.allowsLoadPlanning()  
  Mit diesem Schalter wird die Laderaumplanung in der Tourenplanung aktiviert.
* XFVRP. addOptType (XFVRPOptType) [Default: -]  
  Fügt das übergebene Optimierungsverfahren der Liste zu bearbeitender Verfahren hinzu. Die Reihenfolge des Aufrufs dieser Methode bestimmt auch die Reihenfolge der Bearbeitung.
* XFVRP. clearOptTypes ()  
  Löscht alle bisher hinzugefügten Optimierungsverfahren.
* XFVRPOptType  
  Dieses Enum enthält alle implementierten Verfahren. Bei der Angabe, welches Verfahren vom XFVRP verwendet werden soll, kann eine Instanz aus diesem Enum verwendet werden.

## 4.3 Beispiele

XFVRP vrp = **new** XFVRP();

// Dateneingabe

...

// Welche Parameter

vrp.setMaxRouteDuration(300);

vrp.setCapactiy(100);

vrp.setMetric(new AirMetric());

// Welche Verfahren

vrp.addOptType(XFVRPOptType.*CONST*);

vrp.addOptType(XFVRPOptType.*OPT2Intern*);

// Beschleunigung (optional)

vrp.allowsAcceleratedMetric();

vrp.allowsRoutePlanSplitting();

// Planung starten

vrp.executeRoutePlanning();

// Strukturierter Ergebnisreport  
Report report = vrp.getReport();

**Aufruf einer Tourenplanung**

**Eingabe von Daten**

// Kunden hinzufügen

**for**(Customer c : customerList) {

vrp

.addCustomer()

.setId(c.getId())

.setXlong(c.getXlong())

.setYlat(c.getYlat())

.setDemand(c.getDemand())

.setOpen1(c.getOpenTime())

.setClose1(c.getCloseTime())

.setServiceTime(c.getServiceTime());

}

// Depot hinzufügen

vrp

.addDepot()

.setId(depotID)

.setXlong(depotXlong)

.setYlat(depotYlat);

// Fahrzeug hinzufügen

vrp

.addVehicle()

.setName(vehicleName)

.setNbrOfVehicles(nbfOfVehicles)

.setCapacity1(capacity)

.setMaxRouteDuration(maxRouteDuration)

.setFixCost(fixCostPerVehicle)

.setVarCost(varCostPerDistance);

## 4.4 Metriken

Eine Metrik stellt ein Delegat für den Programmierer dar, um in die internen Abfrageprozesse von Entfernungen und Fahrzeiten einzugreifen. Es gibt verschiedene Arten für die Art wie Distanzinformationen erzeugt und gespeichert werden. Generell gliedern sich die Metriken in Koordinaten- und expliziten Informationsangaben.

Bei Metriken, die auf Koordinatenbasis arbeiten, ist es lediglich notwendig, dass beim Anlegen der Knoten (Depot und Kunden) die Koordinaten-Informationen mitgegeben werden. Anschließend können die folgenden beiden Metriken verwendet werden.

* AirMetric – Liest die Koordinaten zweier Knoten aus dem XFVRPTourModel und errechnet die Entfernung projiziert auf die Erdkugel. Die Fahrzeit korreliert direkt mit der Entfernung und wurde auf 60 KM/h gesetzt.
* EucledianMetric – Aus den Koordinaten zweier Knoten wird der euklidische Abstand errechnet. Diese Metric ist für Test- und Benchmark-Zwecke sinnvoll.

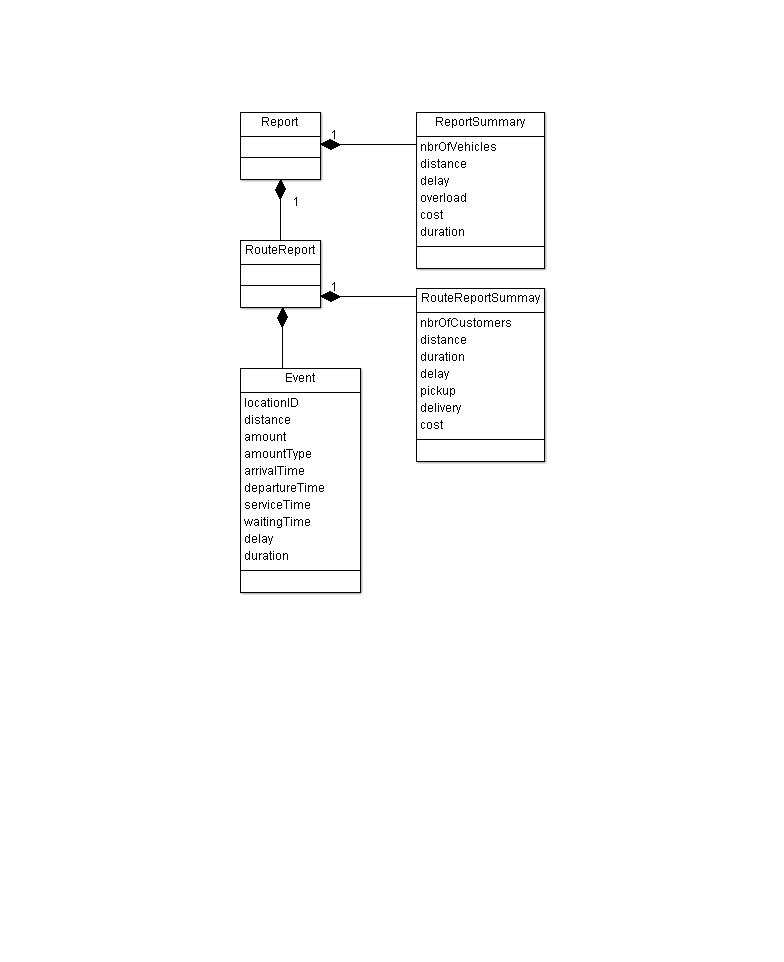
Metriken, die mit vorberechneten Entfernungsinformationen arbeiten, müssen vor ihrer Verwendung im Verfahren initialisiert oder gefüllt werden. Hierbei ist die Metrik mehr ein Datenbehälter.

* **MapMetric** Wird durch die Angabe zweier Knoten-IDs und der Entfernungen gefüllt. Die Befüllung ist sehr einfach. Allerdings ist die Zugriffsgeschwindigkeit eher schlecht, weswegen diese Metrik sich für kleine und schnelle Problemstellungen eignet.
* **DirectMetric** Bei der DirectMetric werden die Entfernungsinformationen als Tupel von Geo-IDs angegeben. Diese müssen bei der Datenaufbereitung (Geokodierung) erzeugt werden und sind für jeden Knoten eindeutig. Zusätzlich ist die Angabe einer oberen Schranke verschiedener GeoIDs anzugeben. Diese Metric ermöglicht den Verfahren einen direkten Zugriff auf die relevanten Informationen, sodass hiermit auch größere Instanzen effizient bearbeitet werden können.  
  Die DirectMetric wird vom XFVRP automatisch aufgebaut, wenn der Schalter allowsAcceleratedMetric() aufgerufen wird. In diesem Fall werden die Daten der vorher vom Benutzer eingegebenen Metric indiziert und in die DirectMetric übertragen.

Es können eigene Metriken durch Implementierung des Metric-Interface erzeugt und verwendet werden. Wichtig ist dabei, dass alle möglichen Kombinationen zweier Knoten einem Entfernungswert zugeordnet werden können. Andernfalls ist die Stabilität des Verfahrens nicht sichergestellt.

## 4.5 Report

Das Ergebnis einer Planung kann als Report-Objekt vom XFVRP-Objekt erfragt werden. Der Report gliedert sich in eine Liste von RouteReports auf, wobei jeder RouteReport eine einzelne Tour beschreibt. Jeder RouteReport enthält Events, die die feinste Beschreibung des Ablaufs einer Tour darstellen. Sowohl der Report selbst als auch die RouteReports enthalten Summary-Objekte, die die Informationen der darunterliegenden Liste zusammenfassen.



## 4.6 Optimierungsverfahren

In der ausgelieferten Version von XFVRP sind verschiedene Optimierungsverfahren für Touren- oder Fahrplanplanung enthalten. Im Folgenden werden die wichtigsten Verfahren genannt, wobei stetig neue hinzukommen. Nahezu alle Verfahren besitzen Komponenten, die sich durch Multithreading parallelisieren lassen, weswegen die Angabe der Thread-Anzahl bei Multi-Prozessorsystemen lohnenswert ist.

Jedes Optimierungsverfahren erbt von der Klasse XFVRPBase und muss damit eine Methode implementieren, die eine Tourenplanungslösung als Eingabe erhält und eine weitere zurück gibt. Da die Optimierungsverfahren über das Benutzerinterface (Enum XFVRPOptType) leicht zugänglich und sehr flexibel austauschbar sein sollen, muss zu Beginn des Optimierungsaufrufes das Optimierungsmodell aus der Klasse XFVRP gesetzt werden.

Die einzelnen Nachbarschaftssuchoperatoren für lokale Nachbarschaftssuchen erben von XFVRPOptImpBase, welche die allgemeine Struktur einer Nachbarschaftssuche enthält. Der Operator ist in diesem Fall leicht austauschbar. Dadurch lassen sich schnell neue Nachbarschaftsoperatoren erzeugen, die durch Kombination bereits vorhandener Verfahren erfolgen kann. Ein Beispiel ist die CONST-Nachbarschaftssuche, welche die Konstruktionsverfahren Savings und Sweep unter einem Aufruf vereint.

### Konstruktionsverfahren

* Trivial:  
  Generell wird zu Beginn eine Stichtourlösung erzeugt, um die korrekte Bearbeitung der späteren Verbesserungsverfahren zu gewährleisten.
* Savings:   
  Initialisiert eine Stichtour-Lösung und verschmilzt pro Iteration zwei Touren miteinander. Diese Verfahren reduziert die Gesamttourlänge recht gut, aber es viele Touren erzeugt.
* Savings-Lamda:  
  Eine modifizierte Variante des Savings bei dem durch einen Parameter gesteuert auch Verschlechterungen zugelassen werden können. Beim Aufruf der Savings-Lamda-Heuristik werden automatisch 5 Parametrisierungen von Lamda durchgespielt und das beste Ergebnis zurückgegeben.
* Sweep: <Deprecated>   
  Der Sweep-Algorithmus ordnet die Kunden in der Reihenfolge ihres Winkels zum Depot den Touren zu. Führt das Hinzufügen des Kunden an das Ende der aktuellen Tour zu einer ungültigen Tour, wird eine neue Tour eröffnet. Diese Greedy-Heuristik wird durch eine 2-Opt\*-Verbesserungsheuristik alle 10 Einfügungen aufgewertet. Das Ergebnis ist vergleichbar mit den Savings-Ergebnissen.  
  Dieses Verfahren wird nicht mehr im Standard-Konstruktionsverfahren eingesetzt, da es mehrere Restriktionen des XFVRP nicht sinnvoll einstellen kann. Hierzu gehören vor allem Zeitfenster.
* Solomon Variante I1:  
  Es werden nacheinander Touren eröffnet und solange mit Sendungen bestückt, bis die Restriktionen es nicht mehr erlauben, die Tour zu erweitern. Die Auswahl bestimmter Planungsparameter wird über Schätzfunktionen durchgeführt, der sehr Zeitfenster affin sind.  
  Reduziert die Tourenanzahl sehr gut, aber die Tourenlängen sind eher hoch.  
  **(Funktioniert nicht mit mehreren Zeitfenstern oder gemischter Be- und Entladung)**

### Verbesserungsverfahren

Verbesserungsverfahren beginnen mit einer konstruierten Lösung und versuchen diese Iterativ zu verbessern. Die hier aufgeführten Verfahren sind durchweg lokale Nachbarschaftssuchverfahren, die ihre Suche in einem lokalen Optimum beenden, wenn keine neuen verbessernden Lösungen mehr gefunden werden können.

* 2-Opt\* (INTRA):  
  Verfahren aus der TSP-Planung bei der Teile der Tour invertiert werden. Besonders beim symmetrischen VRP ohne Zeitfenster ist dieser Operator sehr performant, um schnelle und gute Lösungen zu erhalten.
* 2-Opt:  
  Erweiterung des 2-Opt für das VRP, bei dem auch über Touren hinweg invertiert werden kann. Etwas besser aber deutlich langsamer als 2-Opt.
* Swap:  
  Wählt zwei Kunden auf beliebigen Touren und vertauscht deren Positionen aus. Wird in der Literatur auch Exchange genannt. Kann feinere Verbesserungen als 2-Opt finden.
* Relocate:  
  Versetzt einen Kunden im Tourenplan über Touren hinweg. Für die Arbeit mit sehr engen Zeitfenstern ist dieser Operator aufgrund der geringen Veränderung am Tourenplan sehr effektiv.
* Or-Opt:  
  Verschiebt ein Segment (Knotensequenz) des Tourenplans an eine andere Position. Die Segmente können zwischen 2 und 4 Knoten enthalten.
* 3-Opt:  
  Dieser Operator wählt zwei benachbarte Bereiche des Tourenplans mit beliebiger Länge aus, wobei sich diese nicht überschneiden dürfen. Durch das Invertieren der beiden Bereiche ergeben sich 8 unterschiedliche Fälle, die jeweils überprüft werden. Dieser Operator ist sehr rechenintensiv und nur bei mittleren und kleinen Probleminstanzen sinnvoll einsetzbar.
* SwapSegment:  
  Beim Swap von Segmenten werden zwei Bereiche des Tourenplans mit einander vertauscht. Die Bereiche (auch Segmente) können 2 bis 4 Knoten enthalten, wobei kein Depot vorkommen darf. Den SwapSegment-Operator gibt es in verschiedenen Varianten:
  + Der einfache SwapSegment vertauscht zwei Segmente.
  + Beim SwapSegment mit Invertierung werden pro Vertauschen die vier möglichen Kombinationen probiert, die Segmente in ihrer Reihenfolge ähnlich dem 2-Opt zu invertieren (Nichts invertieren, beide invertieren, nur A invertieren, nur B invertieren)
  + Der „equal“ SwapSegment vertauscht Segmente, wobei beide Segmente die gleiche Anzahl Knoten enthalten müssen. Eine Prüfung mit Invertierung wird nicht vorgenommen.

SwapSegment-Operatoren haben eine Grundkomplexität von O(9n²). Mit Invertierung liegt die Komplexität um Faktor 4 höher. Durch gleich große Segmente reduziert sich der Aufwand auf O(3n²).