KUniKIP README

Fabian Löbel

October 6, 2016

Contents

1 General Information

1.1 About

KUniKIP is an optimization tool for the construction of optimized event schedules for the yearly event KinderUni of the Freie Universität Berlin. For a few days the FU offers a selection of courses with differing dates school classes with children aged between 8 to 12 years can apply for. These classes are then to be assigned to single course dates according to their preferences. However, the courses usually come with capacity and other restrictions and can thus not cover

the huge number of applications. We therefore desire an "optimal" solution under the given circumstances and custom parameters to give as many children as possible the chance to participate. KUniKIP allows us to do just that by modeling the multiple knapsack problem.
Herein the courses are interpreted as knapsacks and the classes as the items we want to distribute over the knapsacks. The classes' weight is given by the amount of pupils in them. The objective function is formulated in respect to what property we seek to maximize, be it the number of pupils, classes, (first-time-)schools or the number of dates with an assignment. KUniKIP will also, to some extend, analyze the given problem to give more insight in the quality of the solution. KUniKIP is written in Perl and utilizes the ZIB Optimization Suite. It is available for Windows and Linux OS and should be adaptable to similar problems. The program and the math behind it are the result of a Diploma thesis by Biliana Boeva that emerged from a joint project of the Zuse Institute Berlin (ZIB) and the Freie Universität Berlin (FU) and was improved and refined by several developers after.

This README was written for KUniKIP Version 1.3. So far Biliana Boeva (original author), Ralf Borndörfer, Gerwin Gamrath, Boris Grimm, Fabian Löbel and Nils Paetsch worked as developers on KUniKIP.

1.2 Mathematical Model: Multiple Knapsack Problem

The Multiple Knapsack Problem (MKP) is the generalization of the Single Knapsack Problem from one backpack to several with differing capacities.
Therefore, given a set of items with a certain value and weight and a set of knapsacks with finite capacities, we seek an assignment of items to the knapsacks which maximizes the total value while the sum of weights of all items in a single knapsack is not greater than its capacity.

1.3 Application to the KinderUni

We interpret the single courses available during the KinderUni as knapsacks and the school classes as items. The resulting objective function can be weighted depending on what value we seek to maximize - be it the number of assigned pupils, (new) schools, classes or occupied course dates.
Furthermore, to reduce frustration, schools having applied the last two years in vain will always be accepted in the third year. There are additional constraints applicable as well:

- a differentiation between course dates that may receive a single class only or multiple
- classes may share a date only if their level does not differ more than by one (always active)

- forbid the joint assignment of two specific classes to a certain date
- statically assign a selected class to a course date
- force the assignment of a class to any course date it was applied for
- define minimum parameters for numbers of pupils, schools, classes and occupied dates
- increase the capacity for certain dates without modifying the base input file

For more information on the mathematical aspects of the problem please refer to the diploma thesis "Veranstaltungsplanung mit Multiple-Knapsack-Methoden" by Biliana Boeva linked on the KUniKIP website.

2 Usage

KUniKIP requires Linux with a Perl distribution and the ZIB Optimization Suite installed (link on KUniKIP website) or Windows with a Perl distribution only (binaries for Windows are contained in the download). To use it provide the necessary input text files in the input folder and run "start_kip.pl".

2.1 Input

All input text files are located in the "input" folder. Each line in an input file requires a single character as an identifier. "c" denotes comments. All entries shall be separated by whitespace. KUniKIP only reads in the specified number of columns, therefore anything can be added to the lines after the required data if it is separated by whitespace from the relevant entries.
Note that all names must not contain any whitespace.

capIncrease.txt:

t dateID increase

This input text file is used to increase the capacity of certain dates without tempering with the main input file. The identifier is "t", it follows the ID of the respective date and a numeral for the capacity increase. Whenever KUniKIP reads in the input the values here a simply added to the capacities of the mentioned dates. If the main input states that date 1337 has a capacity of 40 and there is an entry for 1337 with an increase of two, KUniKIP will do all computations with a capacity of 42 for date 1337. capIncrease.txt is treated as an extension to the main input so that the input data can be modified without changing main.txt. Note that a single entry increases the capacity of

a date only. To increase the capacity of an entire course all of its dates need to be entered here. Due to technical reasons the file always has to contain a single entry - best enter a random date with an increment of 0.

conflicts.txt:

p dateID class1ID class2ID

An entry in this file denotes that the two specified classes must not be assigned to the selected date together. If date 1337 allows multiple class assignments then KUniKIP will never assign both mentioned classes to it, only one or none of them.

doMatch.txt:

v classID

KUniKIP will assign all here listed classes to one of the dates they were applied for irregardless of the impact on the overall solution as long as it remains feasible.

exceptions.txt:

t dateID

By default all dates are single class dates meaning only a single class can be assigned to this date even if there was capacity left for additional classes. Listing a date here turns it into a multiple classes date so that as many classes can be assigned to this as long as the sum of all pupils does not exceed its capacity. Note that this is, again, date-wise and not course-wise.

regions.txt:

b regionName regionID

A list of regions with arbitrary IDs the schools are located in. Currently has no influence on the optimization, however KUniKIP will notify you on the number of regions that have at least one class assigned to a date.

main.txt:

This is the proper input that contains all relevant data. An example:

c	appID	classID	dateID	#Pupils		
a	2286	310	495	14		
c	classID	schoolID	className	classLevel	region	
k	310	235	5b	5	name	
c	dateID	courseID	capacity	courseName	minlvl	maxlvl
t	495	77	30	name	1	5
c	schoolID	school	record			
v	235	name	see below			

As evident "c" denotes a comment and the main input is made up of four sections. While each line has an identifier in respect to its section it is mandatory that no sections are mixed and that the order is exactly as shown. The first section with the identifier "a" lists all applications. Each application requires its own ID (which may be arbitrary but has to be unique), a class ID, a date ID and the amount of pupils. Applications may share classes (resembling that a single class may apply to different dates increasing its chances to be accepted - each class is assigned at most once though) and dates (different applications for the same date). The number of pupils is listed here because a single class may apply with different amount of pupils to different dates. KUniKIP will point out any classes with differing amounts of pupils.
The second section with identifier "k" lists all classes. Any class ID used in the first section has to show up here and any class ID here needs at least on application to be valid. Additional data for each class is the ID of the respective school, a class designation, the class level and the name of the region its school is from. The class level is relevant to determine whether all applications of this class are valid since some courses have restricted age groups. Additionally, for multiple class dates, KUniKIP will only put classes of a similar level together.
The next section with identifier "t" lists all dates. Each date belongs to a course and obviously dates may share courses. It follows the date's capacity indicating how many pupils can be assigned to it, a course declaration and a minimum and maximum class level. Every date that appeared in the first section has to be listed here, there may be dates with no applications, however. KUniKIP will point such dates out.
The last section with identifier "v" denotes all schools. Of most interest is the "record" column. For each school its participation history has to be put here to determine which schools must be assigned this time because they were not in the last two years and which schools are new. The "record" column therefore actually consists of ten columns and the entries are Boolean - 0 for false and 1 for true. The first five columns denote the years the school had at least one class assigned to a date and therefore participated. The fifth indicates last year, the fourth the year before that and the first column the participation five years ago. The next five columns indicate the years the school had at least one application in the same manner. Example:

c	ID	name	'11	'12	'13	'14	'15	'11	'12	'13	'14	'15
v	01	n1	0	0	0	1	0	1	0	0	1	1
v	02	n2	0	0	1	0	0	0	0	1	1	1
v	03	n3	0	0	0	0	0	0	0	0	0	0

Since the time of writing this is 2016, let's say we're optimizing the KinderUni 2016. School 01 therefore applied 2011, 2014 and 2015 and got to participate 2014. School 02 must be assigned this time because it was rejected the last two years. School 03 has not applied in the last five years and is therefore considered to be a new school.

2.2 Output

KUniKIP has several output files located both in the main directory and the "output" folder. The relevant ones are:

output/kip_capacityIncrement.txt:

Contains the result of a capacity increase analysis.

output/kip_eConstraint.txt:

Contains the Pareto-curve for maximizing the amount of pupils with different minimal amounts of new schools.

output/kip_eConstraint2.txt:

Contains the Pareto-curve for maximizing the amount of pupils with different minimal amounts of schools (total).

output/kip_gapAnalyzation.txt:

Lists the result of the gap analysis.

output/kip_main-db.txt:

Solution data in an easy-to-read format for databank tools.

output/kip_main-detail.txt:

Solution data in an easy-to-read format for humans.

output/error.txt:

Error log.

capIncreaseSuggestion.txt:

Based on capIncreaseCandidates.txt (explained later) contains a suggestion for capacity increments. If approved simply replace the contents of input/capIncrease.txt with that.

analysis.txt:

Contains an extensive analysis of the given problem.

2.3 Commands

I you run start_kip.pl you are greeted with a console dialog. It knows the following commands:

h

Shows all available commands and a short explanation (help function).

quit

Ends the program.

a

Analyzes the problem and generates "analysis.txt" in the main directory. In the text file it points out large classes (more than 30 pupils), all dates without any application, all dates remaining free if maximizing the amount of assigned pupils and presents a gap analysis. It furthermore prints the result of maximizing different targets and a (sort of) Pareto-curve on the amount of pupils while setting minimum parameters for the number for assigned schools.
WARNING: Be aware that using this option overrides all files in the "output" directory, so get all files you want to keep to safety.

g

Starts the gap analysis which is part of command "a". The gap is the amount of capacity that can not be filled. It is determined by iterating over all dates and assigning as many pupils as possible to them disregarding any constraints outside the capacity of the specific date. The result is printed to "output/kip_gapAnalyzation.txt".

l

Presents an analysis on what happens if minimum amounts of schools are forced while optimizing the amount of pupils. It takes the total number of schools from the solution maximizing the amount of pupils and iteratively increases it as a minimum parameter for further calculations, computing optimal solutions with those minimum number

of schools. Often, by accepting a slightly lower amount of pupils in the final solution, the number of accepted schools can be increased. This is part of the analysis of "a". The result is located in "output/kip_eConstraint2.txt"

nl

Does the same as "l" except it uses the number of new schools instead of total. It is part of "a" as well, the result is located in "output/kip_eConstraint.txt".

c

Capacity increment analysis that iteratively increases the capacity of selected dates and checks whether optimizing the amount of pupils will yield better results. Entering "c" will prompt KUniKIP to ask for the mode to use - by entering "s" it will ask for a date ID, if neither "s" nor "l" is entered it will consider all dates. In the next step a minimum and maximum capacity increase has to be specified, which KUniKIP will test on the selected dates.
Option "l" requires the file "capIncreaseCandidates.txt" in the main directory that has the same layout as "capIncrease.txt". This file is a list of all dates for which an increase in capacity is possible up to a limit, which is the numeral provided with each date ID. This mode is a bit more refined: Knowing that all specified dates can be increased, KUniKIP will immediately accept the best increment for the currently examined date that yields a higher amount of pupils and continue the analysis with the increased capacity. It then generates a "capIncreaseSuggestion.txt" in the main directory that is based on the analysis and can simply replace "capIncrease.txt" in the input folder after validation by the user. Note that any existing entries in "capIncrease.txt" are considered part of the input data and the analysis will take place on top of those. However, KUniKIP incorporates the existing increases in its suggestion.
Regardless of the selected mode, an analysis of the increments can be found in "output/kip_capacityIncrement.txt".

o

Computes an optimal solution under selectable constraints. You will be prompted to select the value you seek to maximize and may define minimum parameters for all other values. KUniKIP will then determine the optimal solution under these constraints. The result is located in "output/kip_main-detail.txt" for the human eye and in "output/kip_main-db.txt" for data banks.

2.4 Suggested Work Flow

How would one go about using KUniKIP effectively? The suggested work flow begins with setting up the input files. Make sure that there is a single entry in "input/capIncrease.txt" with a random date and an increase of zero but not anything else. Then use command "a" to analyze your problem and verify its validity and the quality of your first solution. You should perform a capacity increment analysis to your liking afterwards and set up an appropriate "input/capacityIncrement" if any courses can and should be increased in capacity. Start another analysis with "a" afterwards and decide on which value you want to maximize and which minimum parameters you desire to set (often that's going to be the optimization of the amount of pupils with some minimum number of assigned schools). Then generate your solution using command "o".
You can of course experiment with KUniKIP's tools and your input further.