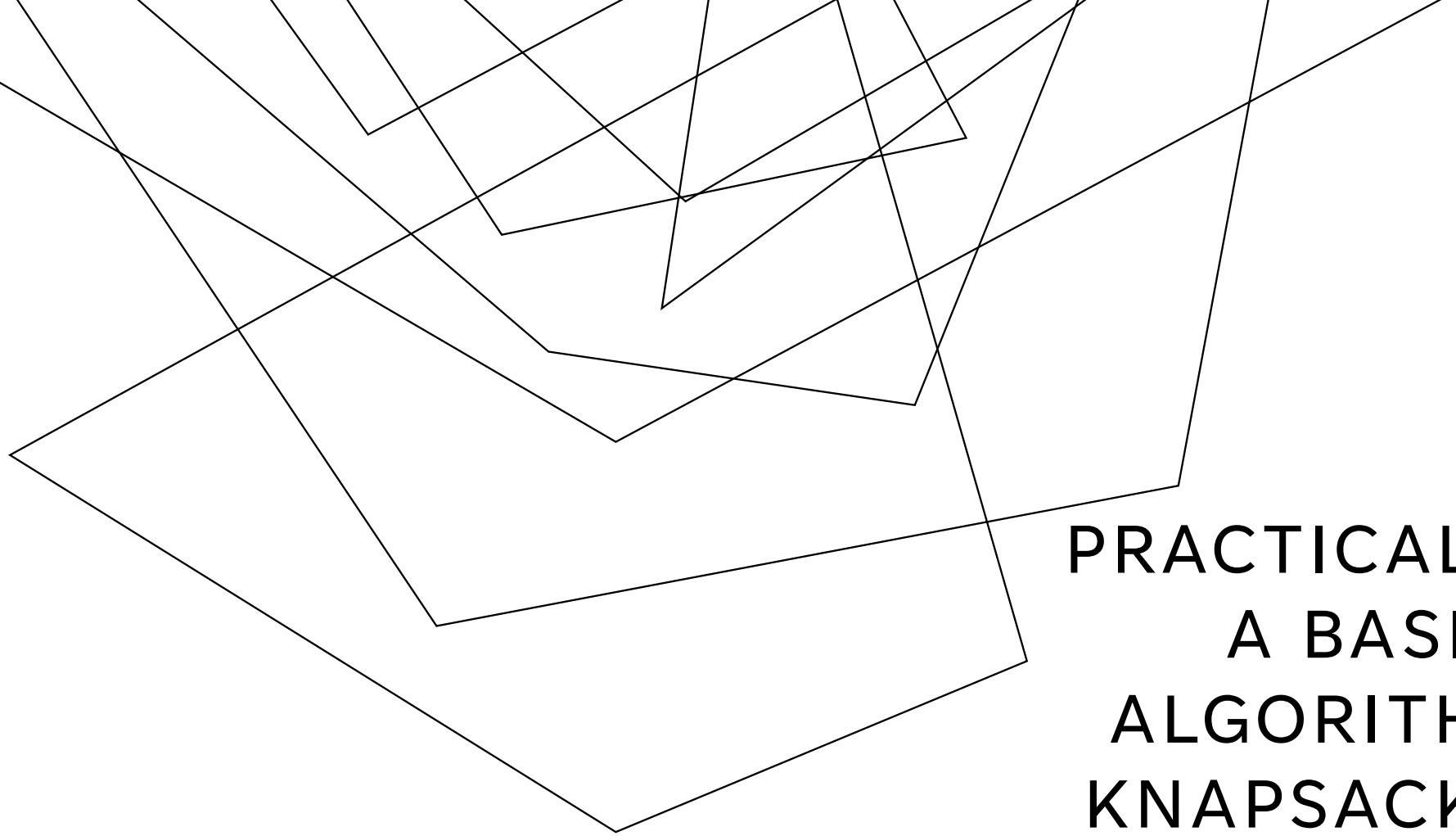


PRACTICAL STUDY OF A BASIC GENETIC ALGORITHM ON THE KNAPSACK PROBLEM

Resolución de problemas con metaheurísticas
Master Universitario en Investigación en Inteligencia Artificial (UIMP)

Author: David Mora Garrido

INDEX

1. Introduction

2. Problem formulation

3. ssGA implementation

4. Methodology

5. Results

6. Conclusions

INTRODUCTION

Multidimensional Knapsack Problem: generalization of the basic binary Knapsack Problem to multiple knapsacks or dimensions.

Given a list of n items, the profit p_j for item j, a number m of dimensions ('knapsacks'), the weight (required capacity) w_{ij} of item j for knapsack i, and a total per-dimension capacity W_i:

Maximize

$$\sum_{j=1}^{n} p_j x_j \quad x_j \in \{0, 1\},$$

subject to

$$\sum_{j=1}^{n} w_{ij} x_j \leq W_i \quad \forall i = 1, ..., m$$

DISAMBIGUATION

Multiple knapsacks - not to be confused with physical dimensions

When an item is selected, it consumes a certain capacity from **all** knapsacks/dimensions

Better problem formulation or real use case by [Petersen, 1967]:

- Each item is an R&D project
- A knapsack is a required resource: R&D, marketing, implementation, etc.
- Each R&D project consumes a certain amount of each resource
- Each resource is limited, there’s a maximum amount or capacity

PROBLEM FORMULATION (FOR A GENETIC ALGORITHM)

- **Problem data**:
  - number of items (n),
  - profit of the items (p_1, ..., p_n),
  - number of knapsacks/dimensions (m),
  - total capacity of each knapsack (W_1, ..., W_m),
  - weight or consumed capacity of each item for each dimension (m times n matrix containing w_{ij} weights).

- **Solution representation**: binary vector of length n, position j is 1 if the j^{th} item is selected, otherwise it's 0.
  - Infeasible solutions exist

$$\exists i = 1, ..., m : \sum_{j=1}^{n} w_{ij} x_j > W_i$$

- Size of the state space depends on the problem instance, upper bounded by 2^n.

SSGA IMPLEMENTATION

Base implementation (NEO investigation group, Universidad de Málaga):

- Selection scheme: binary tournament
- Operators:
  1. Single-point crossover applied given a probability, only one child is generated.
  2. Mutation: per-gene (bit) evaluation given the specified probability.
- Insertion: generated individual replaces the worst individual.

Our modifications/additions:

- New class (`ProblemMKP.java`) with the required attributes to hold the problem instance data and the functionality to compute whether a solution is feasible and depending on that the fitness (**infeasible solutions are tolerated, their fitness is -1**).
  [Chu and Beasly, 1998] proposes different alternatives to deal with these solutions.
- Execution functions to carry out the execution/evaluation/gathering of results (`Exe.java`)

GitHub

https://github.com/damogad/MKP

MKP Public

Pin | Unwatch 1 | Fork 0 | Star 0

main | 2 Branches | 1 Tags | Go to file t | Add file | Code

damogad Merge pull request #1 from damogad/wip — bbae536 · 4 days ago — 13 Commits

input_files	input .txt file with all the MKP instances	w results + comments ...	last week
mkp	code implementation of the ssGA	+ new results + comments ...	last week
results	contains .csv with all the results	+ new results + comments ...	last week
.gitignore		Entropy computation correction + new results + comments ...	last week
README.md	instructions on how to execute (compiled or .jar), used input parameters		st week
mkp.jar		Entropy computation correction + new results + comments ...	last week
results_notebook.Rmd	R notebook executable in RStudio		last week
results_notebook.html	html output of the R notebook		last week

About

Solving the Multidimensional Knapsack Problem with a Genetic Algorithm

Readme
Activity
0 stars
1 watching
0 forks

Releases 1

v1.0.0 Latest
4 days ago

.jar file

Packages

METHODOLOGY

- Employed instances:
  - 3 different different instances, each of a different relative size (small, medium, large), extracted from [Petersen, 1967]. 3rd, 5th and 7th problems in http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/mknap1.txt
  - Fixed population size: 100, 100 and 200, respectively.

- Types of studies:
  1. Fixed number of evaluations: evaluate obtained fitness
  2. Reach the optimal fitness: evaluate the needed number of evaluations

- 25 candidates/combinations of these hyperparameters:
  - Crossover probability: {0.6, 0.7, 0.8, 0.9, 1.0}
  - Mutation probability: x / n, where x is {1, 2, 3, 4, 5} and n is the number of items (number of genes). Meaning: average expected number of mutated genes per individual.

METHODOLOGY

- Metrics:
  - Required number of evaluations (2nd study). 1st study: 1000, 10000 and 1000000, respectively
  - Fitness of the best individual in the last population. Relevant for both studies, as we may obtain an 'unfinished' execution in the 2nd study (50000000 max. evaluations).
  - Average fitness in the complete last population
  - Measure of the variety in the last population genotype: Shannon Entropy

$$\frac{1}{n}\sum_{i=1}^{n} H(x_i)$$

where

$$H(x_i) = \begin{cases} 0 & \text{if } p_i^1 = 0 \vee p_i^1 = 1 \\ -p_i^1 \log_2(p_i^1) - p_i^0 \log_2(p_i^0) & \text{otherwise} \end{cases}$$

METHODOLOGY

- The ssGA is executed 30 times for each combination of hyperparameters (crossover+mutation)

- Statistical hypothesis testing:
  - Parametric methods:
    - Shapiro-Wilk normality test
    - ANOVA
  - Non-parametric methods:
    - Kruskal-Wallis test
    - Wilcoxon rank-sum test (Mann-Whitney U test) with [Benjamini and Hochberg, 1995] correction to avoid inflating the Type I error

- Tested variables:
  - 1st study: relative deviation from the target fitness -> (target – obtained)/target
  - 2nd study: number of evaluations

- **0.02** level of significance (α)

RESULTS

Processing/evaluation of the results in R: https://github.com/damogad/MKP/blob/main/results_notebook.Rmd
Already generated output (html): https://github.com/damogad/MKP/blob/main/results_notebook.html

Practical study of a basic genetic algorithm on the Knapsack problem

David Mora Garrido

2024-01-03

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

Introduction

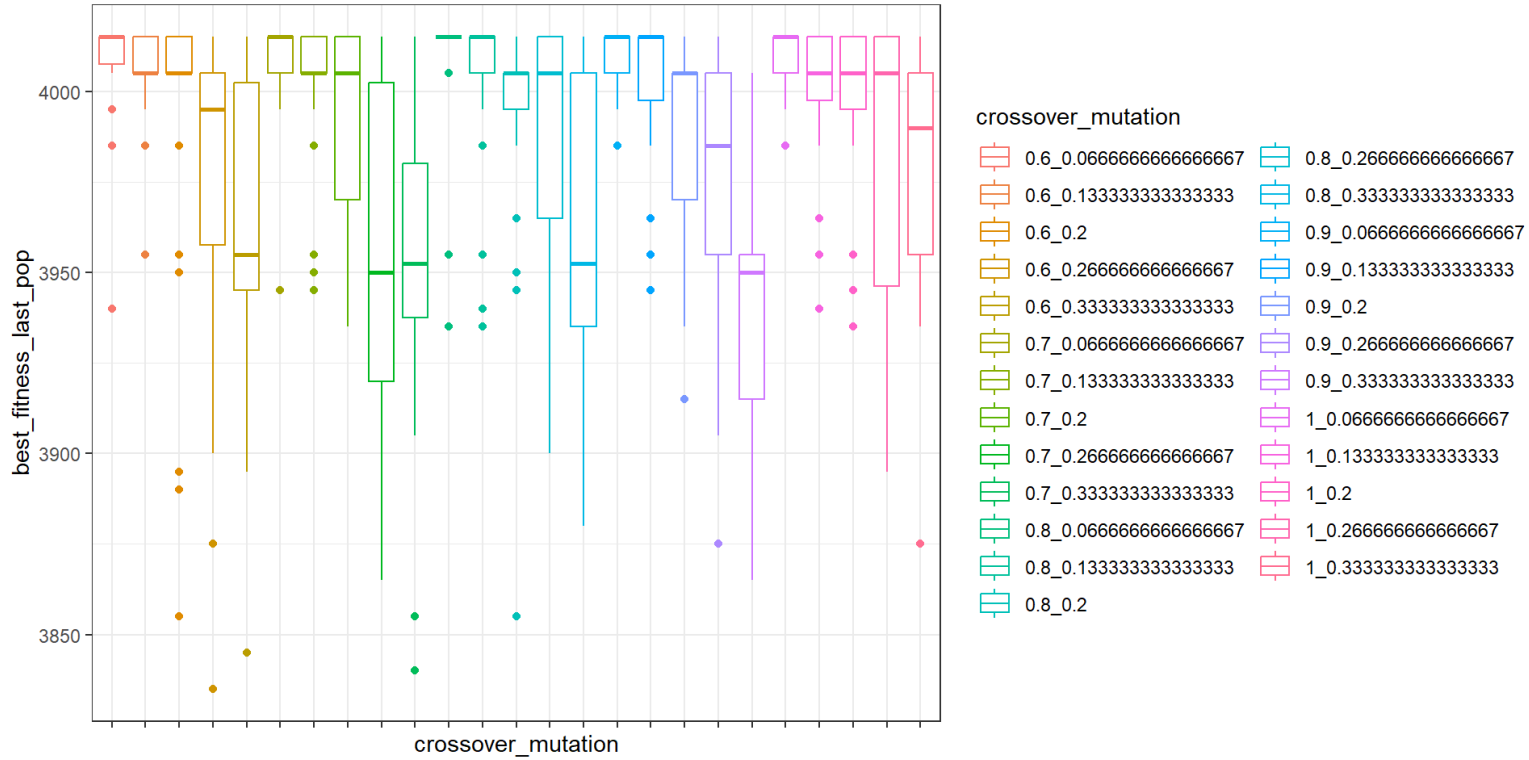
First we load the results `.csv` file:

```
df <- read.csv('./results/mknap1.csv')
head(df)
```

	problem_index <int>	population_size <int>	crossover_probability <dbl>	mutation_probability <dbl>	search_optimal <chr>
1	2	100	0.6	0.06666667	false
2	2	100	0.6	0.06666667	false
3	2	100	0.6	0.06666667	false
4	2	100	0.6	0.06666667	false

RESULTS: 1ST STUDY

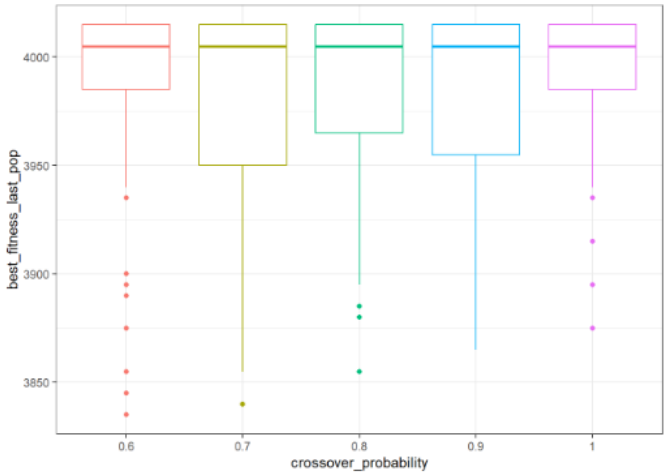
Graphical depiction of the results for each combination of crossover and mutation probabilities (small instance):



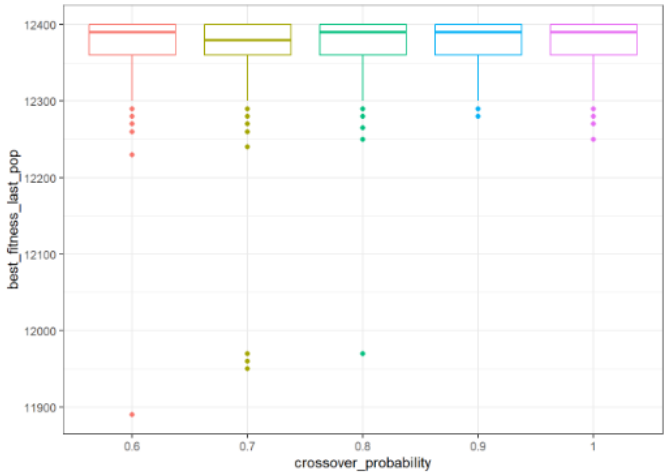
See the similarities/pattern?

RESULTS: 1ST STUDY

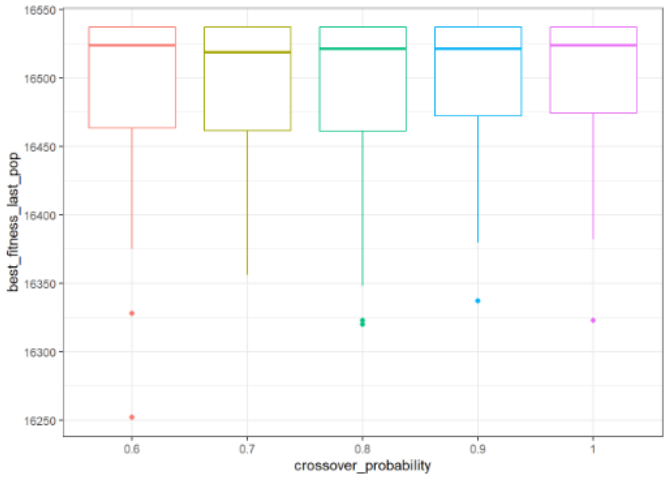
Crossover probability does not seem to have a big effect...



(a) Small instance



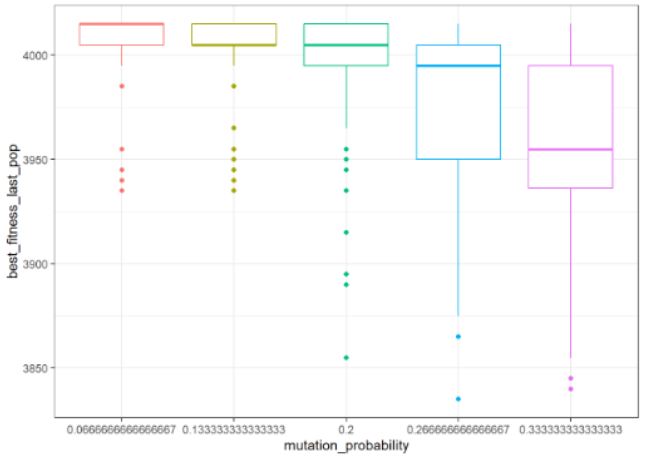
(b) Medium instance



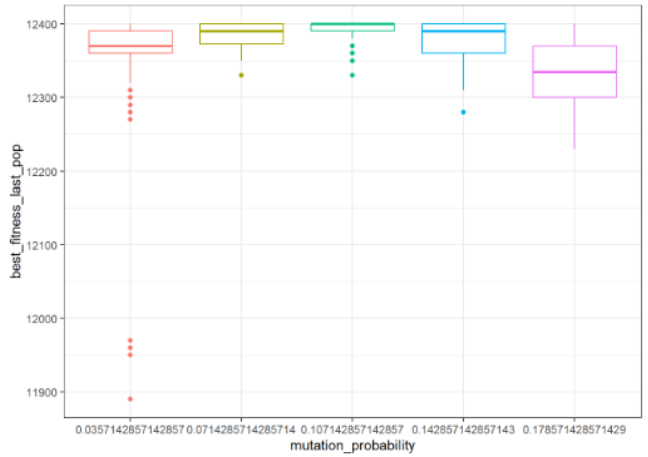
(c) Large instance

RESULTS: 1ST STUDY

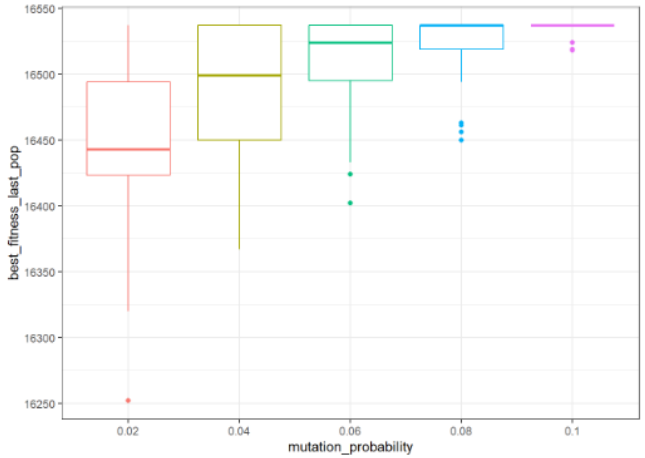
However, the mutation probability **does** seem to influence the best obtained fitness



(a) Small instance



(b) Medium instance



(c) Large instance

RESULTS: 1ST STUDY

Verification of assumptions: **hypothesis tests**

- Shapiro-Wilk normality test: null hypothesis rejected for most combinations, for the 3 instances

```
## [1] "0.6,0.0666666666666667"
##
##  Shapiro-Wilk normality test
##
## data:  df_study1_small[df_study1_small$crossover_mutation == combination, "optimal_deviation_best_last_pop"]
## W = 0.50817, p-value = 6.545e-09
##
## [1] "0.6,0.133333333333333"
##
##  Shapiro-Wilk normality test
##
## data:  df_study1_small[df_study1_small$crossover_mutation == combination, "optimal_deviation_best_last_pop"]
## W = 0.69397, p-value = 1.276e-06
##
## [1] "0.6,0.2"
##
##  Shapiro-Wilk normality test
##
## data:  df_study1_small[df_study1_small$crossover_mutation == combination, "optimal_deviation_best_last_pop"]
## W = 0.61915, p-value = 1.26e-07
##
## [1] "0.6,0.266666666666667"
##
##  Shapiro-Wilk normality test
##
## data:  df_study1_small[df_study1_small$crossover_mutation == combination, "optimal_deviation_best_last_pop"]
## W = 0.78276, p-value = 3.23e-05
##
```

...

RESULTS: 1ST STUDY

Verification of assumptions: **hypothesis tests**

- Kruskal-Wallis test on the equality of the deviation from the optimal among all the combinations of crossover + mutation probabilities: null hypothesis rejected for the 3 instances → **the distribution of at least one combination is not the same as the rest**

```
##
##  Kruskal-Wallis rank sum test
##
## data:  optimal_deviation_best_last_pop by crossover_mutation
## Kruskal-Wallis chi-squared = 350.83, df = 24, p-value < 2.2e-16
```

- Next step: Wilcoxon rank-sum test between every pair (300 total tests) with BH correction → **significant number of pairs that are not equal between them** (154, 160 and 186, respectively)

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  df_study1_large$optimal_deviation_best_last_pop and df_study1_large$crossover_mutation
##
##          0.6,0.02 0.6,0.04 0.6,0.06 0.6,0.08 0.6,0.1 0.7,0.02 0.7,0.04 0.7,0.06
## 0.6,0.04 0.00020  -        -        -        -       -        -        -
## 0.6,0.06 7.0e-07  0.05867  -        -        -       -        -        -
## 0.6,0.08 8.4e-09  0.00037  0.02361  -        -       -        -        -
## 0.6,0.1  2.1e-09  2.7e-05  0.00247  0.34628  -       -        -        -
## 0.7,0.02 0.76371  0.00033  4.8e-07  2.9e-09  1.1e-09 -        -        -
## 0.7,0.04 0.00105  0.58696  0.01109  5.2e-05  2.9e-06 0.00319  -        -
## 0.7,0.06 1.0e-05  0.41108  0.36796  0.00250  0.00018 1.1e-05  0.14141  -
## 0.7,0.08 3.9e-09  0.00013  0.01239  0.70392  0.57575 2.2e-09  1.4e-05  0.00099
## 0.7,0.1  3.9e-09  0.00010  0.00903  0.71568  0.61041 1.5e-09  9.9e-06  0.00082
## 0.8,0.02 0.82923  0.00055  4.2e-07  5.7e-09  1.5e-09 0.86481  0.00419  1.8e-05
## 0.8,0.04 0.00631  0.31714  0.00377  1.2e-05  1.0e-06 0.01176  0.66471  0.04676
## 0.8,0.06 1.3e-05  0.15922  0.81985  0.06778  0.00669 1.8e-05  0.05278  0.50741
## 0.8,0.08 1.1e-08  0.00080  0.05056  0.89273  0.28266 5.8e-09  8.8e-05  0.00499
## 0.8,0.1  1.5e-09  9.8e-06  0.00055  0.17742  0.70365 9.5e-10  1.2e-06  5.0e-05
## 0.9,0.02 0.40027  0.00319  6.8e-06  2.5e-08  2.7e-09 0.53254  0.01827  0.00015
## 0.9,0.04 0.00011  0.95870  0.05835  0.00015  9.3e-06 0.00029  0.58917  0.37796
## 0.9,0.06 3.3e-06  0.19380  0.67285  0.00664  0.00059 1.6e-06  0.04812  0.65952
## 0.9,0.08 5.4e-08  0.00520  0.26624  0.51832  0.10920 6.4e-08  0.00081  0.03636
```

...

RESULTS: 1ST STUDY

Verification of assumptions: **hypothesis tests**

But we saw that the crossover probability did not seem to affect the best obtained fitness significantly...

- Grouped by crossover probability (only):
  - Kruskal-Wallis test: null hypothesis not rejected for the 3 instances → **obtained fitness can be considered equal regardless of the crossover probability (on the tested set of values)**

```
##
##  Kruskal-Wallis rank sum test
##
## data:  optimal_deviation_best_last_pop by crossover_probability
## Kruskal-Wallis chi-squared = 1.6399, df = 4, p-value = 0.8016
```

- Grouped by mutation probability (only):
  - Kruskal-Wallis test: null hypothesis rejected for the 3 instances → **at least one mutation probability yields a different fitness**
  - Wilcoxon rank-sum tests (5 mutation probabilities = 10 pairs to be tested) → 10, 9 and 10 tests are rejected, respectively, so **there's a significant influence of the mutation probability on the fitness**

RESULTS: 1ST STUDY

Numerical results (mean/median/std best fitness, mean/median deviation from optimal) for each mutation probability:

Mutation	Avg. fitness	Median fitness	Std fitness	Avg. deviation	Median deviation
0.06666667	4009.167	4015	13.13861	0.001452885	0.00000000
0.13333333	4002.267	4005	19.57250	0.003171440	0.00249066
0.20000000	3992.900	4005	30.94306	0.005504359	0.00249066
0.26666667	3975.800	3995	39.90374	0.009763387	0.00498132
0.33333333	3958.467	3955	41.71631	0.014080531	0.01494396

Mutation	Avg. fitness	Median fitness	Std fitness	Avg. deviation	Median deviation
0.03571429	12353.93	12370	82.36355	0.0037150538	0.0024193548
0.07142857	12386.27	12390	16.28384	0.0011075269	0.0008064516
0.10714286	12393.67	12400	12.81865	0.0005107527	0.0000000000
0.14285714	12379.00	12390	24.15693	0.0016935484	0.0008064516
0.17857143	12333.83	12335	42.50707	0.0053360215	0.0052419355

Mutation	Avg. fitness	Median fitness	Std fitness	Avg. deviation	Median deviation
0.02	16447.07	16443	53.684399	0.0054379069	0.005684223
0.04	16491.17	16499	43.521408	0.0027715628	0.002297877
0.06	16511.47	16524	33.840069	0.0015436093	0.000786116
0.08	16527.70	16537	18.816225	0.0005623753	0.000000000
0.10	16534.15	16537	6.587956	0.0001725424	0.000000000

Smaller instance, smaller number of expected mutated genes by chance per individual

Medium instance, medium number of expected mutated genes by chance per individual

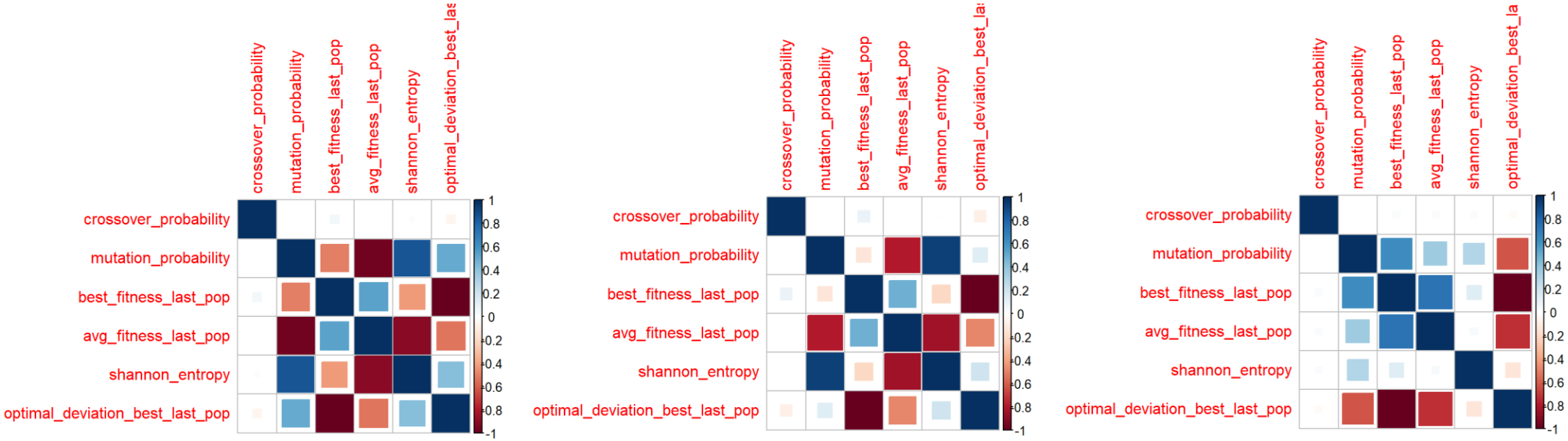
Larger instance, larger number of expected mutated genes by chance per individual

RESULTS: 1ST STUDY

Small and medium instances: negative correlation between the average fitness of last population and the entropy

Large instange: (almost) no correlation between the aforementioned variables

→ Greater num_items/knapsacks may increase the diversity in the genotype?

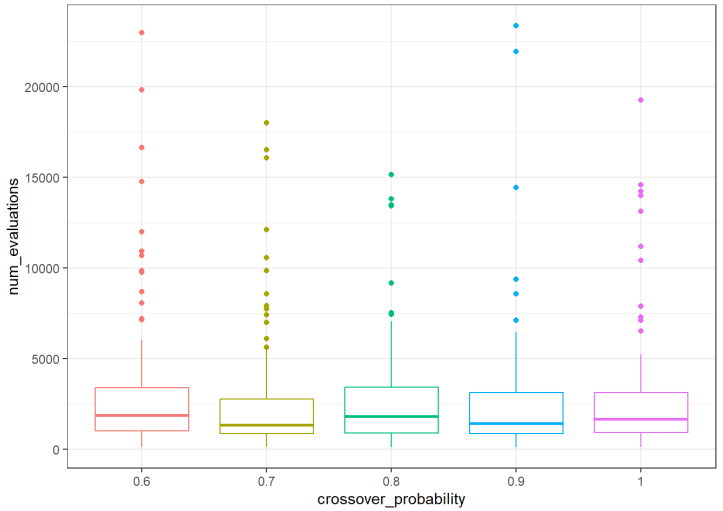


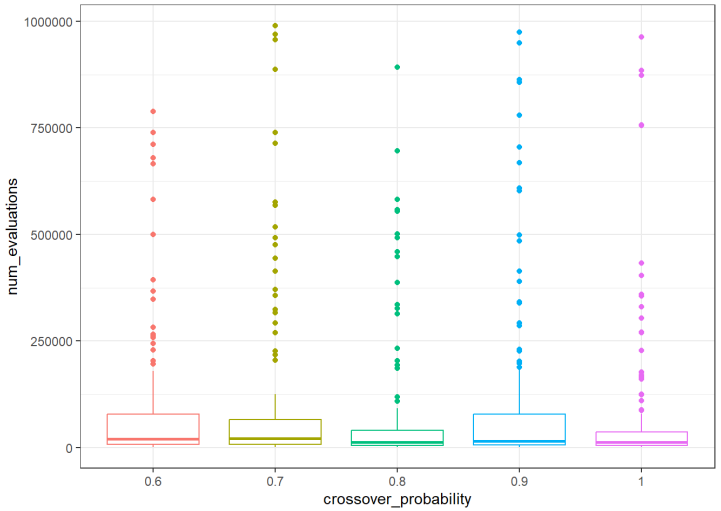
RESULTS: 2ND STUDY

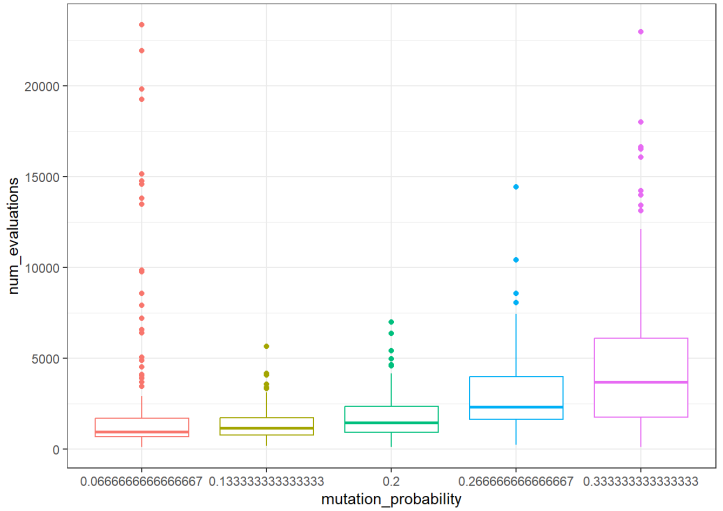
Small and medium problem instances: same results than 1st study

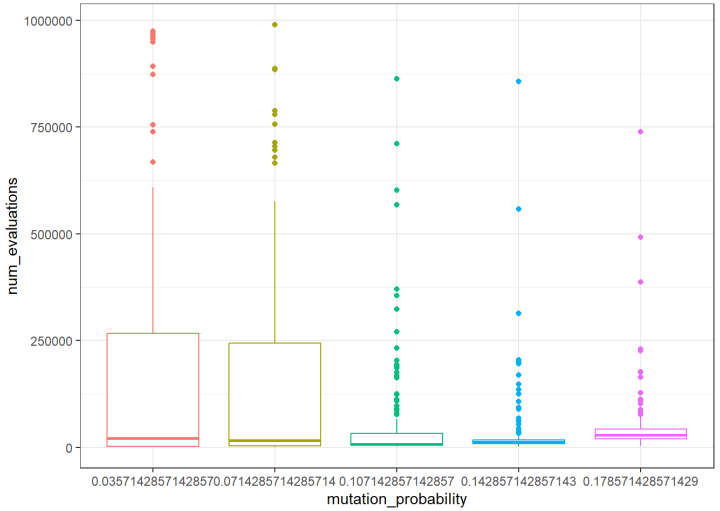
- Different crossover probabilities yield similar number of evaluations
- Different mutation probabilities yield statistically significant differences in the number of evaluations
  - For a smaller instance, a smaller number of expected mutated genes on avg. yields less evaluations
  - Analogously for the medium-sized instance (medium number in the tested set of values)

(tested again with Kruskal-Wallis + Wilcoxon rank-sum tests)









RESULTS: 2ND STUDY

Large instance: several peculiarities

- Optimal fitness was not always reached (‘unfinished’ execution)
- Modification: max limit of 50000000 evaluations when targeting a fitness value
  - If not reached, metrics on current state written as result and execution is reset
  - Reset until obtaining a ‘finished’ execution
- We cannot use the unfinished executions for the study (435)
- ‘Stripped’ results for tests, but number of such executions checked:
  - Crossover probability: evenly distributed, except for one value
  - Smaller mutation probability, greater number of unfinished executions

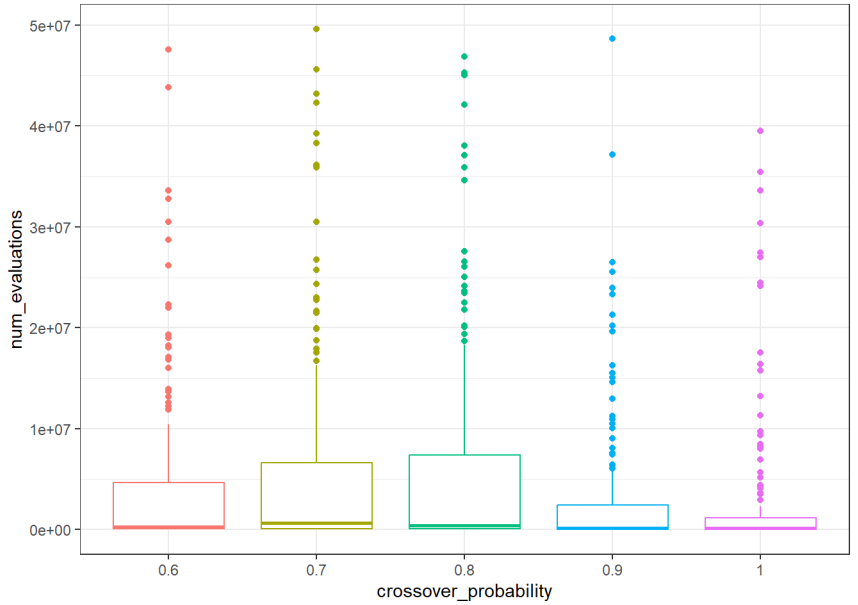
crossover_probability <dbl>	optimal_not_reached <int>
0.6	87
0.7	70
0.8	70
0.9	88
1.0	120

mutation_probability <dbl>	optimal_not_reached <int>
0.02	291
0.04	53
0.06	38
0.08	30
0.10	23

RESULTS: 2ND STUDY

Large instance: several peculiarities

- 'Stripped' results:
  - Different crossover probabilities yield a different number of evaluations



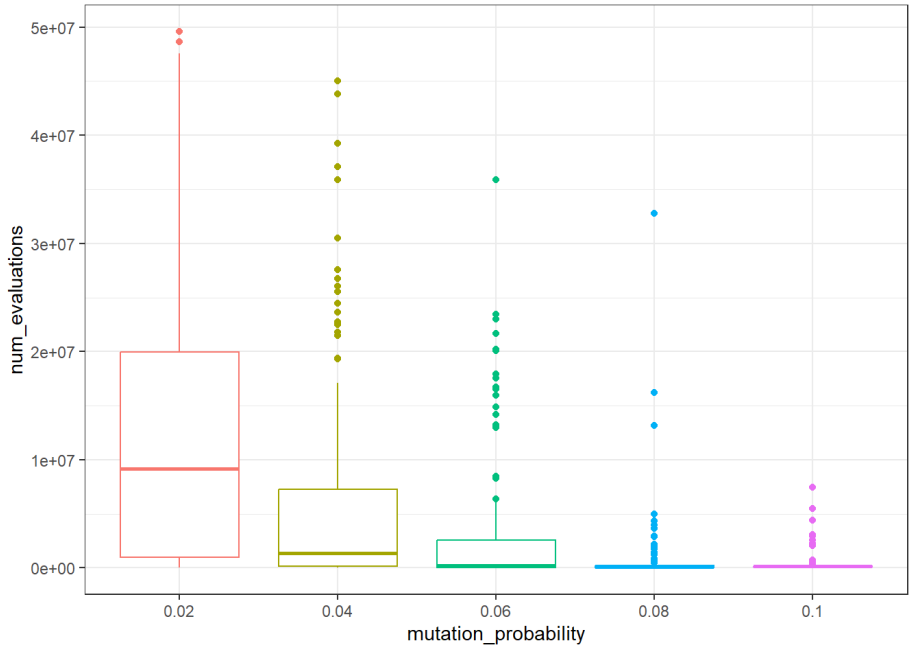

```
##
##  Kruskal-Wallis rank sum test
##
## data:  num_evaluations by crossover_probability
## Kruskal-Wallis chi-squared = 24.147, df = 4, p-value = 7.462e-05
```

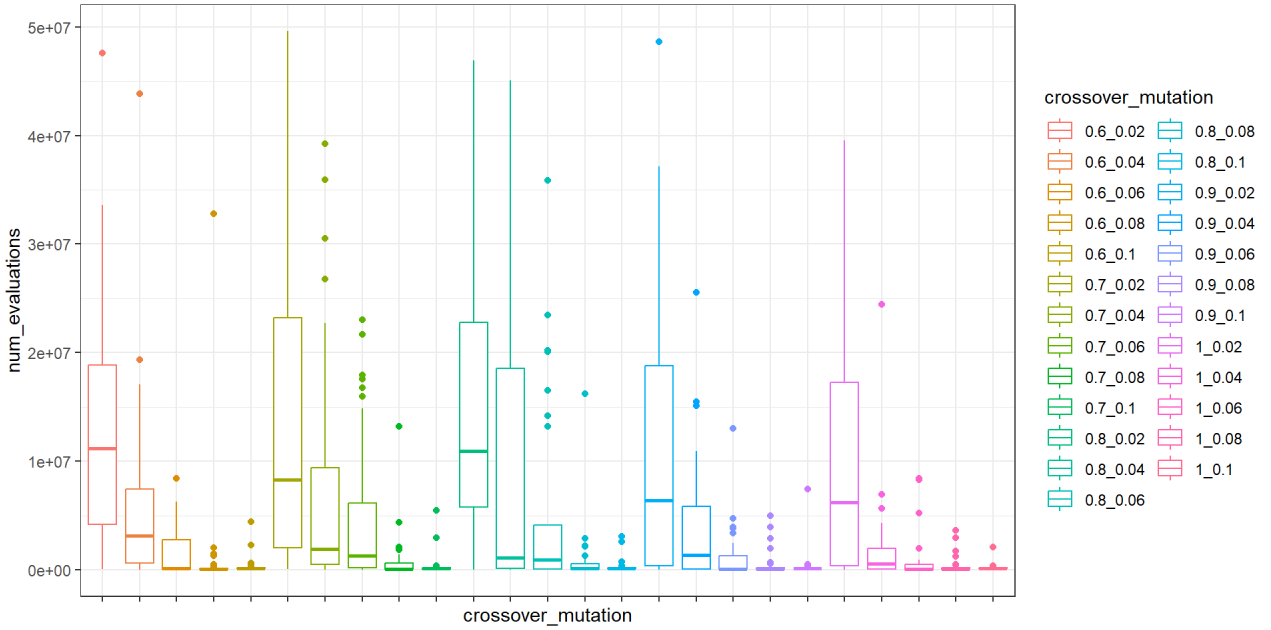
```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  df_study2_large_optimal$num_evaluations and df_study2_large_optimal$crossover_probability
##
##     0.6    0.7    0.8    0.9
## 0.7 0.5832 -      -      -
## 0.8 0.6801 0.7338 -      -
## 0.9 0.0308 0.0050 0.0142 -
## 1   0.0045 0.0013 0.0036 0.5832
##
## P value adjustment method: BH
```

RESULTS: 2ND STUDY

Large instance: several peculiarities

- ‘Stripped’ results:
  - Different mutation probabilities yield a different number of evaluations (already expected)
  - Thus, as Kruskal-Wallis test for the equality of the number of evaluations of all the crossover+mutation combinations is rejected, given the Wilcoxon test results and the balance between median, mean and variance, best combination is (c=1.0, m=0.08).





RESULTS: 2ND STUDY

Large instance: fairer study

- Relaxation of the problem: trying to get as less unfinished executions as possible
  - 99.5% of the optimal: still many unfinished (77)

crossover_probability <dbl>	optimal_not_reached <int>
0.6	17
0.7	14
0.8	17
0.9	15
1.0	15

mutation_probability <dbl>	optimal_not_reached <int>
0.02	58
0.04	15
0.06	4
0.08	1
0.10	0

  - 99% of the optimal: only 3 unfinished

RESULTS: 2ND STUDY

Large instance: fairer study (99%)

- 'Back to normal':
  - No significant differences among the tested crossover probabilities
  - Kruskal-Wallis rejected for the results grouped by mutation probabilities, with Wilcoxon test and numerically aggregated results, best result is obtained with 0.06 (3 expected mutated genes per individual on average)

- Mutation probability: 0.06 or 3/50 (now) vs 0.1 or 5/50 (then)
  - May be caused by the problem being a 'relaxed' version
  - The original problem may be tailored to be hard to get close to the optimal (100%)
  - Thus, it may be easier to escape a local optimum with a smaller probability than before

Crossover probability	Avg. evaluations	Median evaluations	Std. evaluations
0.6	680515.9	11079	4653050
0.7	486655.8	9272	4327903
0.8	333479.7	9462	2977488
0.9	516886.7	9792	4518431
1.0	167130.7	8862	1031928

Mutation probability	Avg. evaluations	Median evaluations	Std. evaluations
0.02	2033935.33	6102.0	8116662.536
0.04	80863.77	6397.5	891338.375
0.06	9025.78	8383.5	3034.596
0.08	12522.85	11697.5	4776.126
0.10	18869.42	18235.5	6585.632

CONCLUSIONS

- Most results are very similar when comparing the two studies → fixed number of evaluations is better, time is guaranteed to be bounded

- Overall not significant effect of the (tested) crossover probabilities → nature of the problem and number of existing constraints, easier to solve an unfeasible solution by mutating than by applying the crossover operator

Improvements/future work
Applying a ‘repair’ operator or a heuristic in order not to add unfeasible solutions to the population.

- [Chu and Beasley, 1998] approach: greedy algorithm based on a DROP phase and an ADD phase
- [Shah, 2020]: Lagrangian multipliers + ‘greedy crossover’

REFERENCES

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society. Series B (Methodological), 57(1):289–300.

[Chu and Beasley, 1998] Chu, P. and Beasley, J. (1998). A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics, 4(1):63–86.

[Petersen, 1967] Petersen, C. C. (1967). Computational experience with variants of the balas algorithm applied to the selection of rd projects. Management Science, 13(9):736–750.

[Shah, 2020] Shah, S. (2020). Genetic algorithm for the 0/1 multidimensional knapsack problem.