Project 6: Model for the Spread of Infectious Diseases

Carlo von Carnap

Summer Semester 2023

Final Project *Computergestütztes wissenschaftliches Rechnen*,
Salvatore R. Manmana

Supervisor: Emily Klass
Submission Date: 08.08.2023

Contents

1 Introduction

Infectious disease modelling describes the process of applying mathematical models to predict the future state of an epidemic. Recently it has surged in popularity during the COVID-19 pandemic and its influence on recent policies has been overwhelming, despite its mediocre success in projecting reality. The difficulties when modelling infectious diseases range from the overcomplex real situations to finding suitable assumptions to keep an appropriate level of accuracy. Generally it can be distinguished between macroscale models such as the SEIR-model[1] that apply differential equations on a population level, and microscale models that have individuals as their smallest unit.

This model uses a quadratic grid of size $L \times L$ and stochastic methods to examine the spread of an infectious agent on a microscale. Each grid node is occupied by an immobile person and can be in one of the states Susceptible S, Infected I or Recovered R. Later, the forth state Vaccinated V is introduced, that permanently excludes the respective people from taking any of the three other states. For a person in one of the three active states S, I and R, the following three transitions are possible,

$$\text{Susceptible } S \xrightarrow{p_1} \text{Infected } I \xrightarrow{p_2} \text{Recovered } R \xrightarrow{p_3} \text{Susceptible } S.$$

With that, we have the three transition probabilities p_1, p_2 and p_3:

- p_1: a susceptible person getting infected by a direct infected neighbor
- p_2: an infected person turning recovered
- p_3: a recovered person returning susceptible

Vaccinated people V are set at the beginning of the simulation, with p_4 being the probability that a spot is occupied by one. The other three states are initialized with the same likelyhood, as the simulation steps progress by each updating L^2 randomly chosen nodes according to the above mentioned rules.

Goal was now to use this model to examine the influence of the transition probabilities on the infection rates averaged over all simulation steps. This included varying the $S \to I$ turover rate p_1 for diffrent combinations of $p_2\,(I \to R\,)$ and $p_3\,(R \to S\,)$ as well as using different vaccination rates p_4 at initialization. For all simulations, different grid sizes were used to observe potential statistical inaccuracies.

Further, the time evolution of the infection rates was looked at to stress the validity of the preceding time averaging. Therefor a quantity of 20 infection rate samples was generated over simulation time with mean and standard deviation calculated for each timestep.

2 Methodology

2.1 Pseudorandom Number Generator (PRNG) MT19937

For the generation of pseudorandom numbers the 1997 developed PRNG Mersenne Twister MT19937 was chosen.[2][3] The PRNG has a

3 Implementation

3.1 Integration of external Libraries

Alongside the C standard library headers, the `gsl_rng.h` of the GNU Scientific Library (GSL) as well as the two numeric libraries `cvc_numerics.h` and `cvc_rng.h` were included in the project. Of the standard library, the header files `stdio.h`, `stdlib.h`, `tgmath.h` and `time.h` were used, the GSL header `gsl_rng.h` was only accessed to generate pseudorandom nubers.

The `cvc_numerics.h` of the two own libraries provides mainly numerical methods for differentation, integration or to solve differential equations, while the `cvc_rng.h` offers functions requiring PRNGs like for instance Monte-Carlo integrators or tools helpful to analyse stochastic datasets such as calculating mean or standard deviation of given arrays.

3.2 Structure and Workflow of the Main Program

3.2.1 Generation of Random Numbers by a Static PRNG

For the generation of pseudorandom numbers the in Subsection 2.1 described Mersenne Twister MT19937 was used. It was implemented statically to generate uniformly distributed numbers in the interval $[0, 1)$ and can therefore be accessed globally by the respective functions.

Overall the PRNG was used in two different ways, once to obtain probabilities, other to generate random integers. The former was accomplished by checking if the uniform was smaller than the respective probability, the latter by multiplying the uniform by the largest desired integer and then casting it to `int`.

3.2.2 Implementation of the Modelling Grid

The above mentioned grid itself was realized as a $(L + 2) \times (L + 2)$ heap section of integer values, with L being the sidelength of the quadratic grid where the actual spread of the infection takes place. While this section is technically one-dimensional, it will for simplicity reasons be here refered to as a two-dimensional structure of the given shape. Inside the grid, the following integer values were used to model the different states of the people within the simulation:

- `0`: this person is Susceptible S to the infection
- `1`: the person is Infected I
- `2`: the person is Recovered R and currently not susceptible
- `-1`: the person is Vaccinated V and does not participate in the spread

The grid was implemented with a supporting edge of ghosts at the top, bottom, left and right border, that are neither infectious nor subject to any updates of the grid — they will permanently take the value `0` and are irrelevant for the later visualization and analysis of the data.

3.2.3 Functions acting on the Modelling Grid

In the main program all changes in the grid are facilitated by functions outside the main one. They entirely work with or change the grid in-place and all take the grid itself as well as its length $L + 2$ as arguments. Inside the functions, the grid is refered to as `grid` and its legth as `length`, while the above probabilities are passed in form of the double array `probabilities` having p_1, p_2, p_3 and p_4 as its entries. Excluding the ones to calculate the infection rate and its time average, none of the functions does have a return value. Overall the following functions can be called from the main:

- `void print_grid`: prints the passed grid as terminal ouput
- `void grid_init`: initializes the grid applying the vaccination probability
- `void update_node`: updates the given node selected by its $L \times L$ grid coordinates `row_i` and `column_j` according to the turnover probabilities
- `void grid_update_linear`: calls the `update_node` function for each node in order of their grid position
- `void grid_update_stochastic`: calls `update_node` for L^2 randomly chosen nodes
- `double ratio_infected`: calculates the ratio of infected individuals with respect to the total grip population for a given grid
- `double average_ratio_infected`: applies `T` simulation steps to the grid by calling the stochastic grid updater with passed turnover probabilities and averages the above infection rate over each simulation step

3.2.4 Structure of the Main Function and General Workflow

The main function is divided into sections for each individual analysis step, where every section is an enclosed space in order to prevent the variables from interfering with each other. The first section uses the terminal input

Overall a running time of around three minutes should be expected after the interactive part is finished.

4 Results and Discussion

4.1 Model for the Spread of Infectious Diseases

The model was implemented in the previously described way and operates mainly by

4.2 Ratio of Infected People in the Grid averaged over Time

Aim was now to analyse the

4.3 Vaccinated People without Participation in the Spread

4.4 Time Evolution of the Expected Ratio of Infected People

While previously the average of the ratio of infected individuals has been taken over time, the focus should now be layed on the time development of th einfection rate $\langle I \rangle_t$ for $N = 20$ samples. As grid size $L = 64$ was chosen for appropriate balance between running time and accuracy, the number of simulation steps again was set to $T = 1000$.

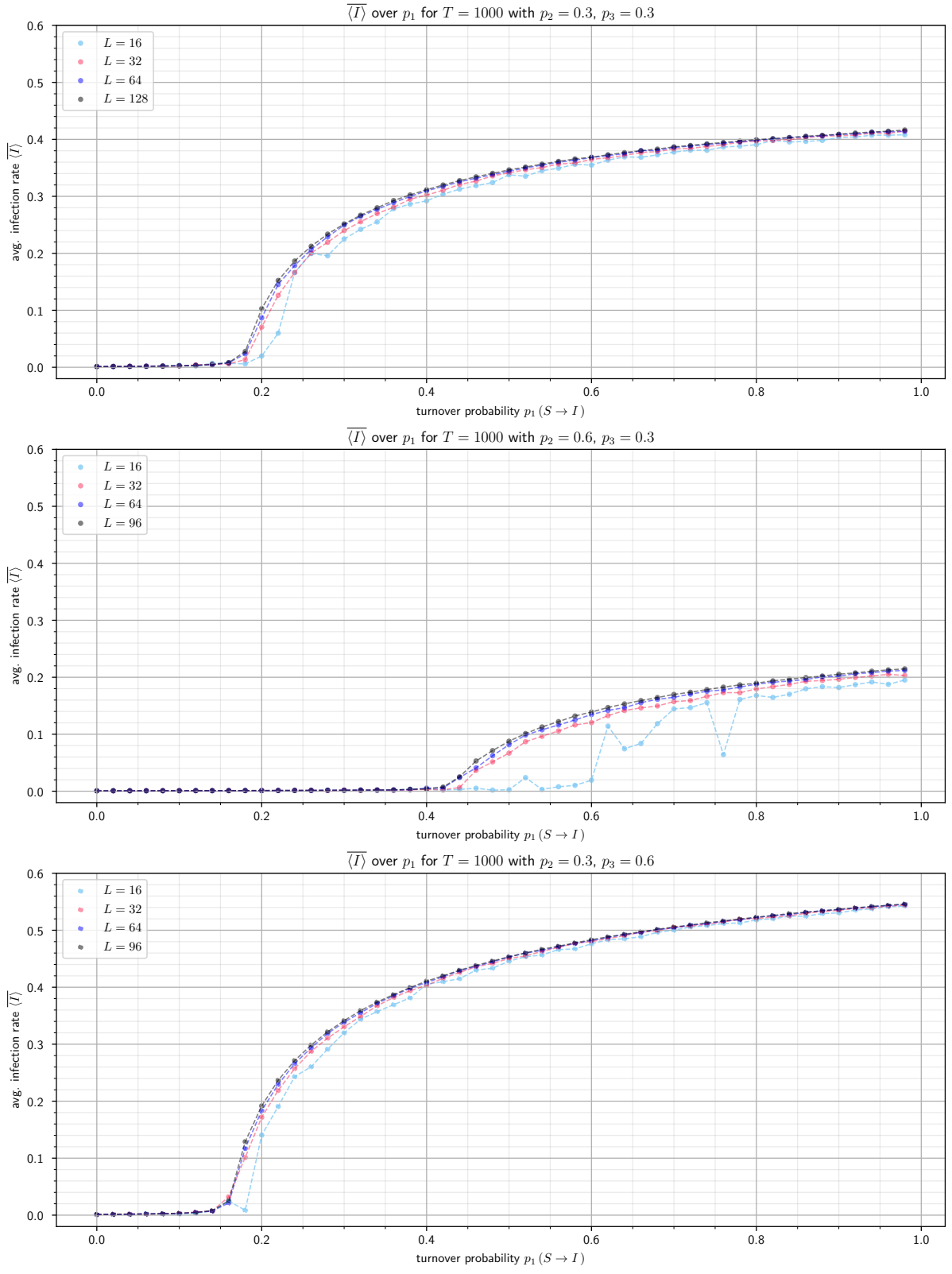

Figure 1: Plots showing the time-averaged infection rates $\overline{\langle I \rangle}$ in dependence of the turnover probability $p_1\,(S \to I)$ for the different grid sizes $L = 16$, $L = 32$, $L = 64$ and $L = 128$ with $T = 1000$ simulation steps each. The upper plot shows has now been generated using the turnover rates $p_2\,(S \to I) = 0.3$ and $p_3\,(S \to I) = 0.5$, the middle one using $p_2 = 0.6$ and $p_3 = 0.3$, and the lower one $p_2 = 0.3$ and $p_3 = 0.6$.

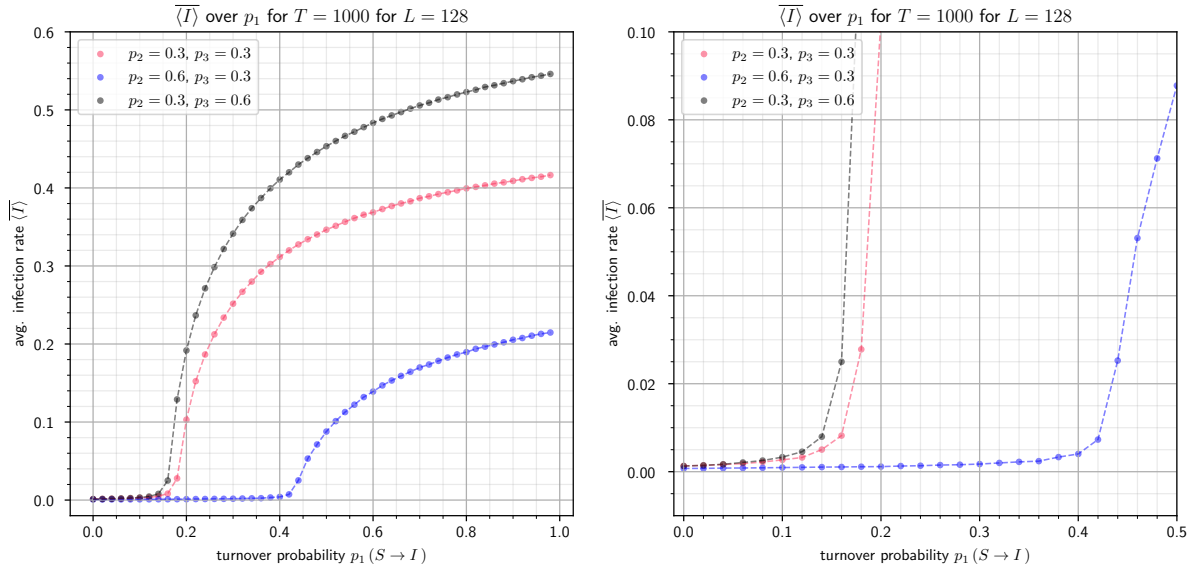

Figure 2: Detailed view of the time-averaged infection rates $\overline{\langle I \rangle}$ over $p_1\,(S \to I)$ for $L = 128$ and $T = 1000$ simulation steps. On the left the different trends for the respective choices of $p_2\,(S \to I)$ and $p_3\,(S \to I)$ can be observed, while on the right the critical values of p_1 for $\overline{\langle I \rangle}$ approaching zero are visible.

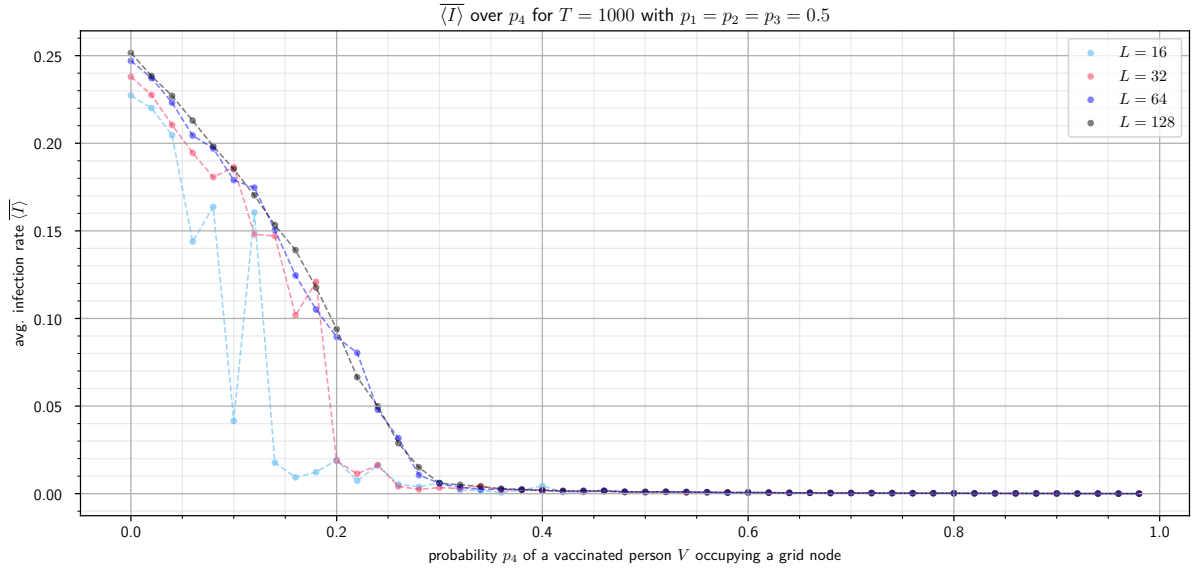

Figure 3: Display of the time-averaged infection rates $\overline{\langle I \rangle}$ depending on the vaccination rate p_4 at the beginning of the simulation and constant turnover probabilities $p_1 = p_2 = p_3 = 0.5$. The simulation was performed over $T = 1000$ simulation steps and th respective grid sizes $L = 16$, $L = 32$, $L = 64$ and $L = 128$.

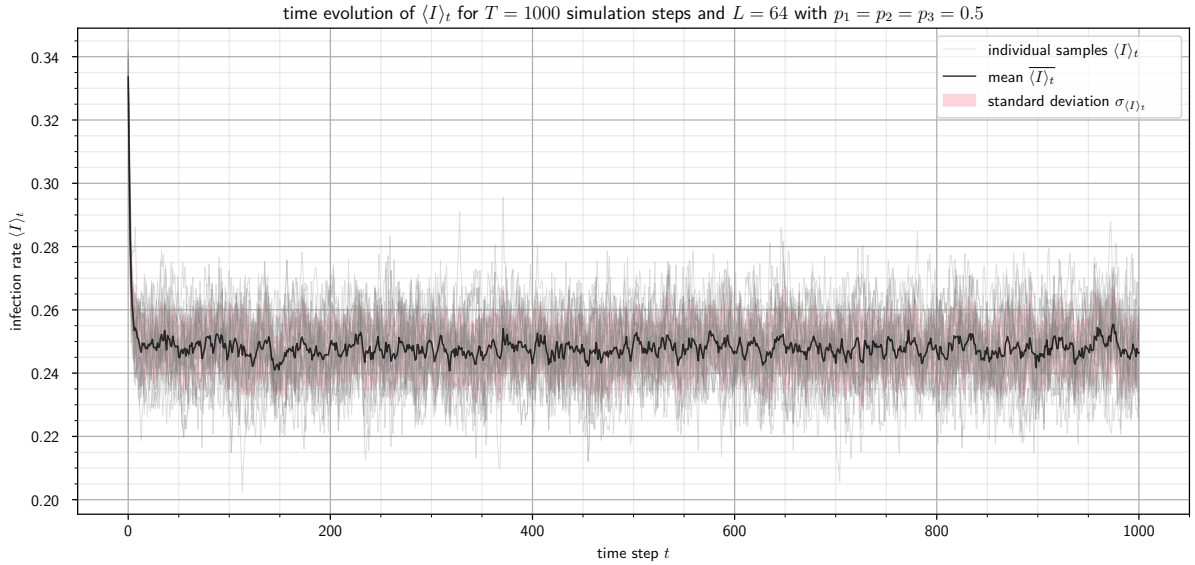

Figure 4: Time evolution of the infection rate $\langle I \rangle_t$ over each of the $T = 1000$ simulation steps for a total of 20 samples. At each timestep mean $\overline{\langle I \rangle_t}$ and standard deviation $\sigma_{\langle I \rangle_t}$ of the infection rate were calculated and are also displayed in the plot. The grid size was chosen as $L = 64$, the turnover probabilities as $p_1 = p_2 = p_3 = 0.5$, no vaccinated induviduals were used.

5 Supplementary Materials

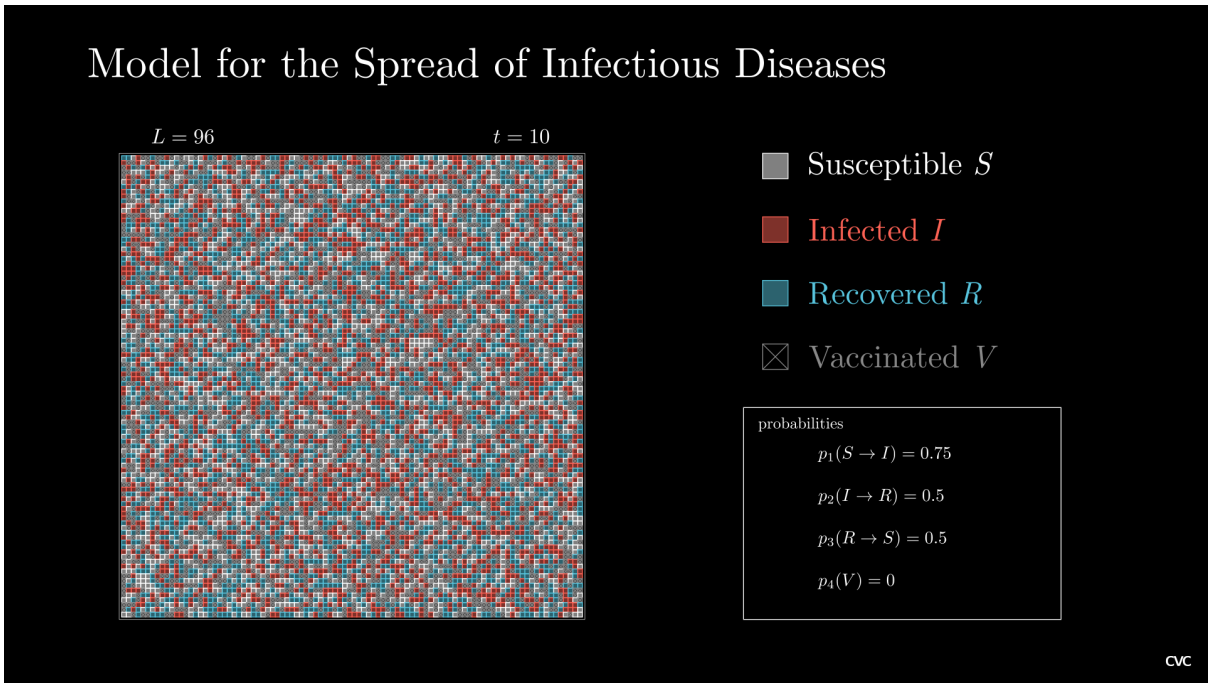

Figure 5: Frame $t = 10$ of an animated simulation of the infectious disease model for the grid size $L = 96$. The grid was initialized with a vaccination rate $p_4 = 0.25$ and progressed with the turnover probabilities $p_1 = 0.75$ as well as $p_2 = p_3 = 0.5$. The total animation as well as variations in the probabilities can be found under `/soi_animations`.

Bibliography

[1] Cornelis P. Dullemond. *EpiDemo. SEIR-Type models.* URL: `https://www.ita.uni-heidelberg.de/~dullemond/software/epidemo/seirmodels.html` (visited on 07/31/2023).

[2] *educative. What is Mersenne Twister?* URL: `https://www.educative.io/answers/what-is-mersenne-twister` (visited on 07/29/2023).

[3] Makoto Matsumoto and Takuji Nishimura. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator". In: *ACM Trans. Model. Comput. Simul.* 8.1 (1998). ISSN: 1049-3301. DOI: `10.1145/272991.272995`. URL: `https://doi.org/10.1145/272991.272995`.