Lecture 1: Introduction

Optimization problem/program

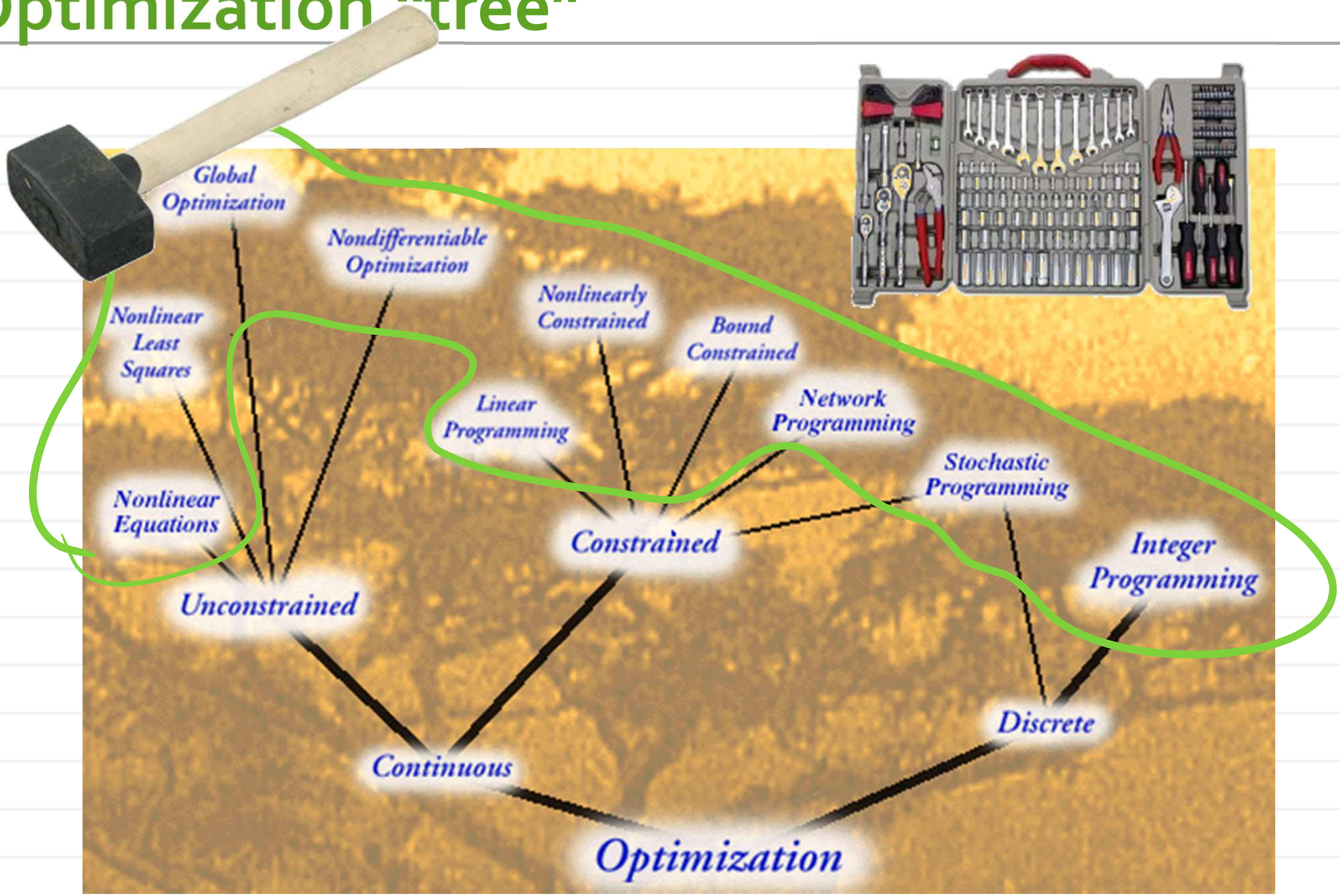
"minimize" objective

$$\min f_0(x)$$

$$\text{s.t.: } x \in D$$

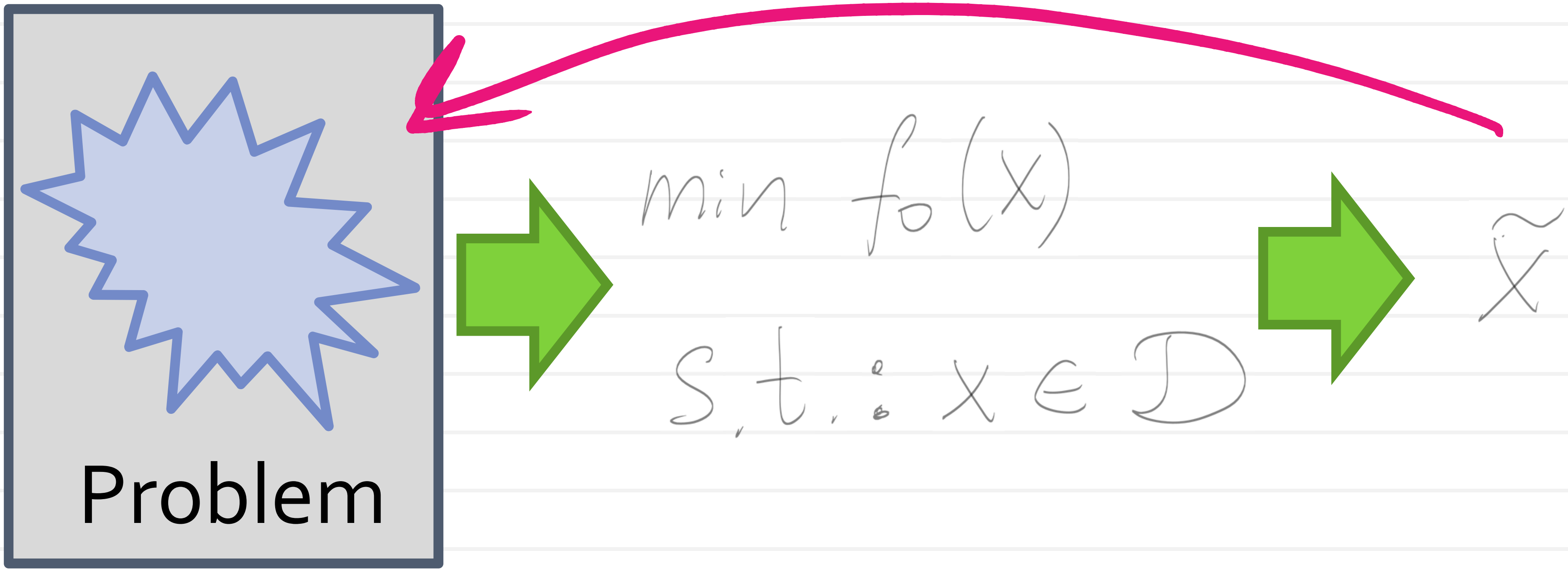
"subject to" domain

Optimization "tree"



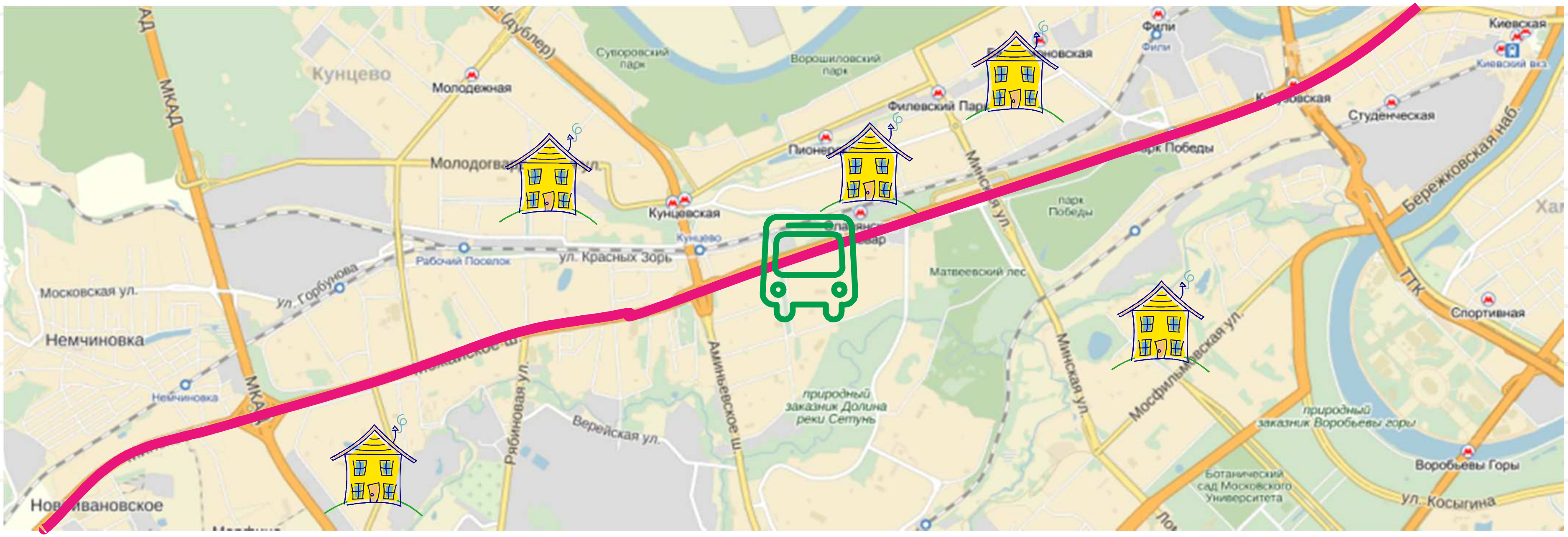
Source: NEOS optimization guide

This class (a one slide overview)



1. What are the standard tools?
2. How to build more powerful tools?
3. How to map real problems to optimization?

Planning the bus stop

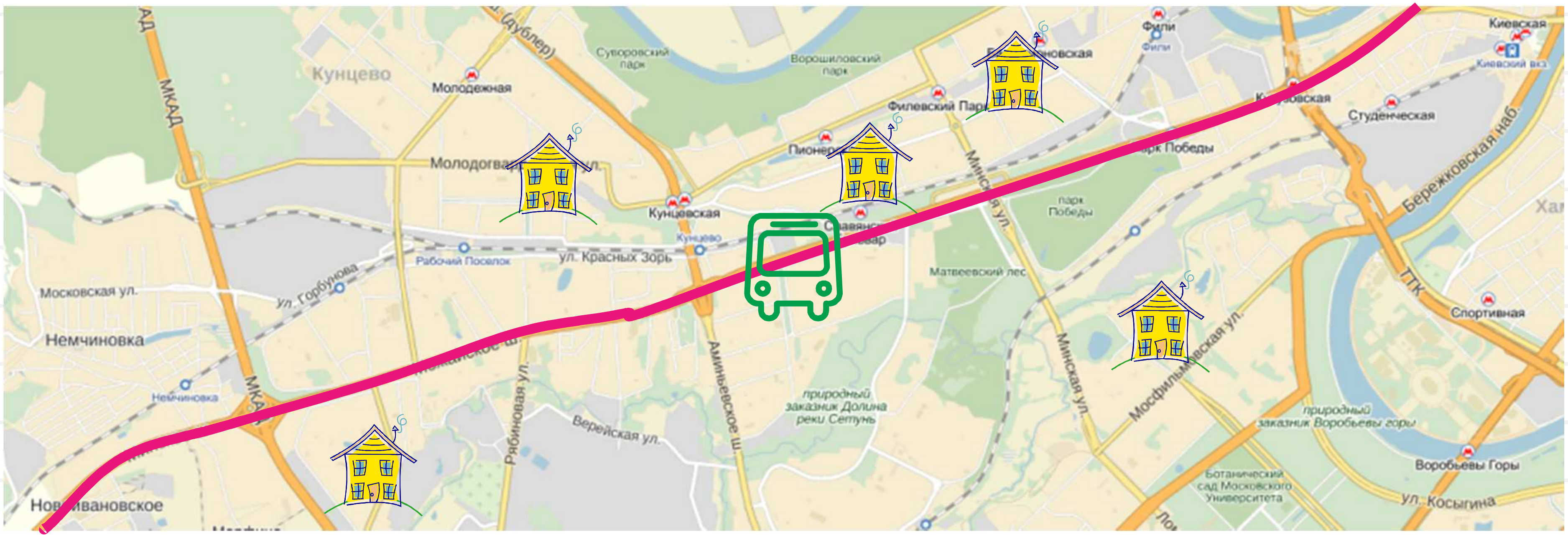


Where should Skoltech place the shuttle stop for its MS students?

Potential factors to consider:

- Where do students live?
- Whether it is possible/convenient to use certain places as a stop?
- Commute distance from the stop to Skoltech.

Planning the bus stop

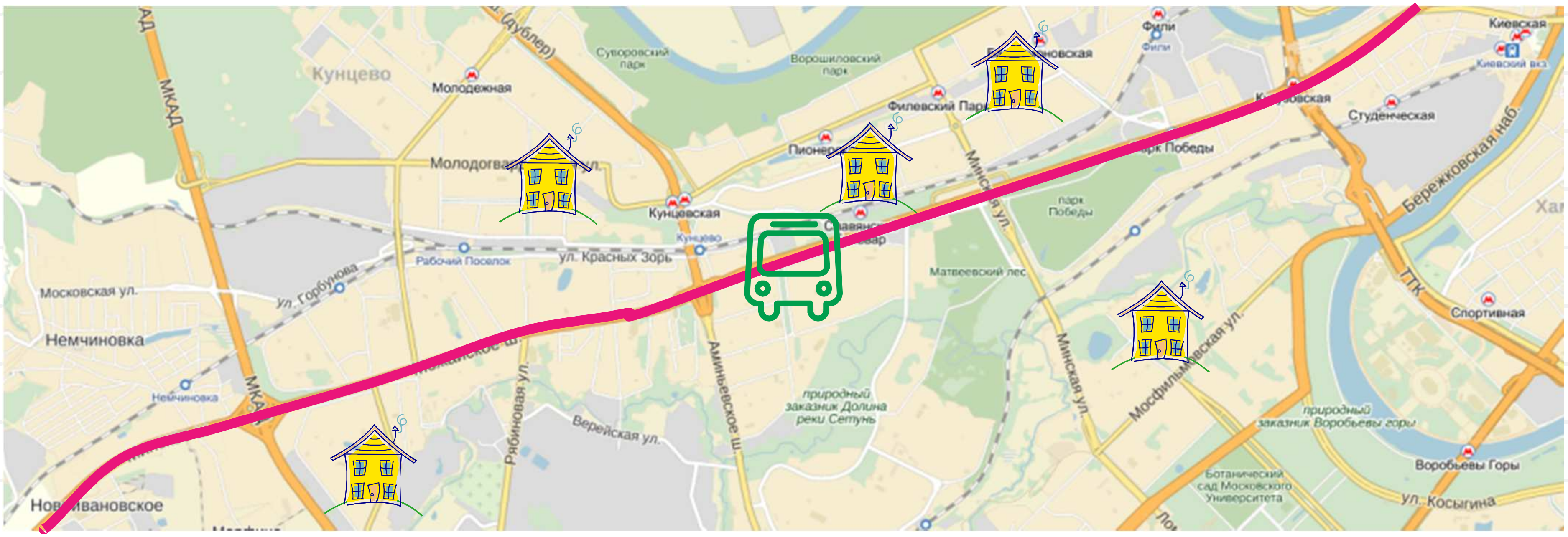


Where should Skoltech place the shuttle stop for its MS students?

Steps for a principled solution:

1. Model each factor with a feasibility function of *x* (where *x* is a position of the stop).
2. *Optimize* the sum of the factors (easy in this case – exhaustive search).

Planning the bus stop

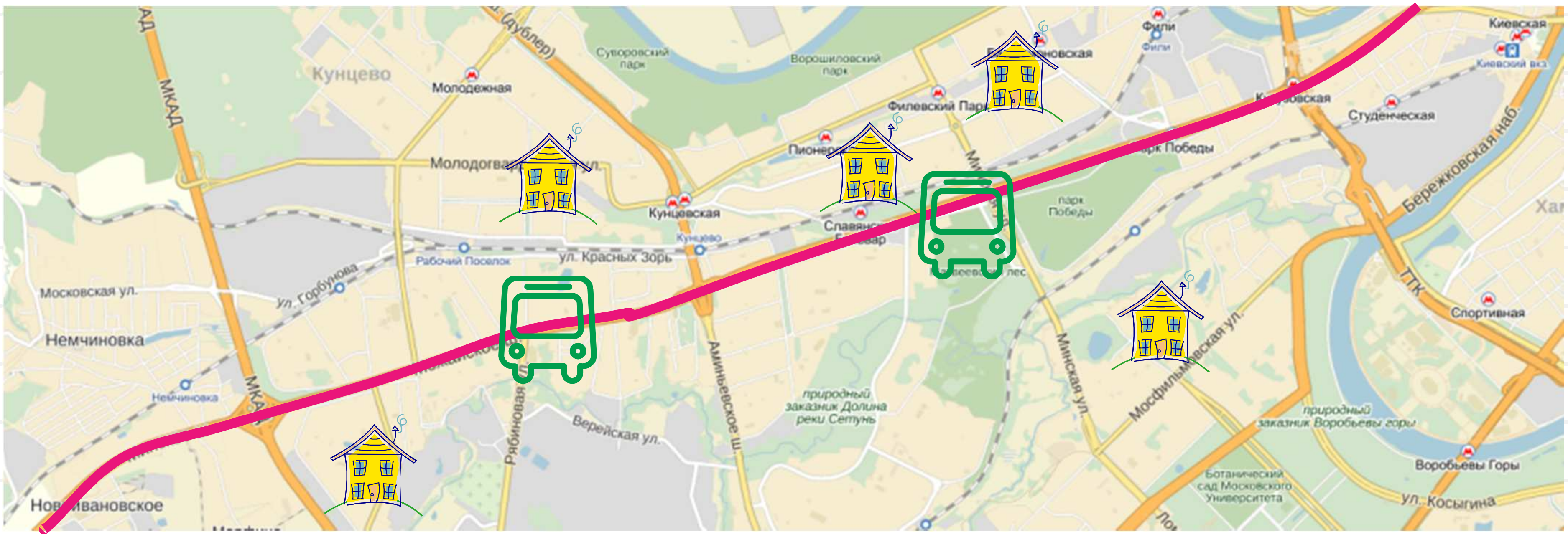


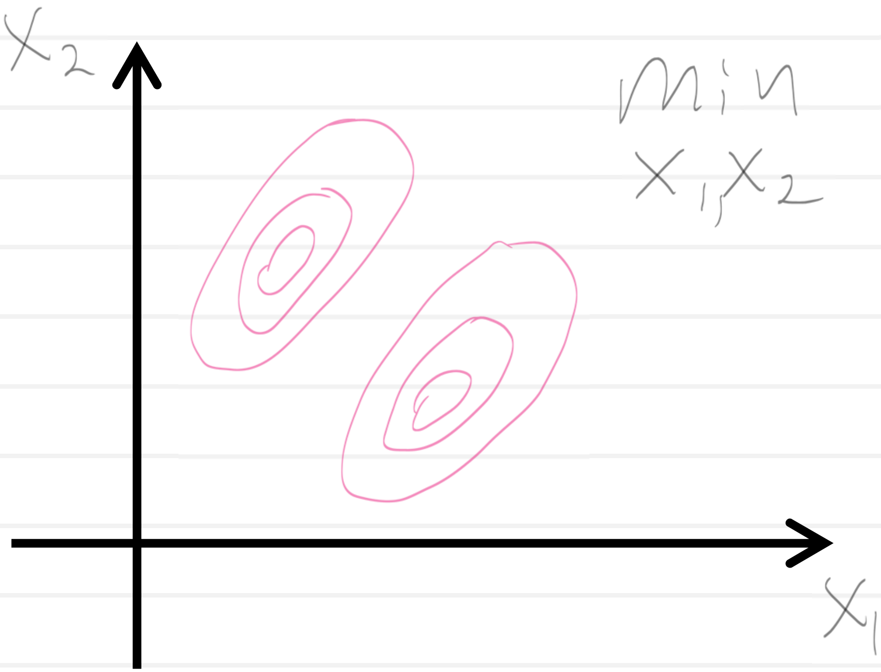
$$\min_{x} \left(u(x) + \sum_{s} \left(d(y_s, x) + d(x, Sk) \right) \right)$$

Steps for a principled solution:

1. Model each factor with a feasibility function of *x* (where *x* is a position of the stop).
2. *Optimize* the sum of the factors (easy in this case – exhaustive search).

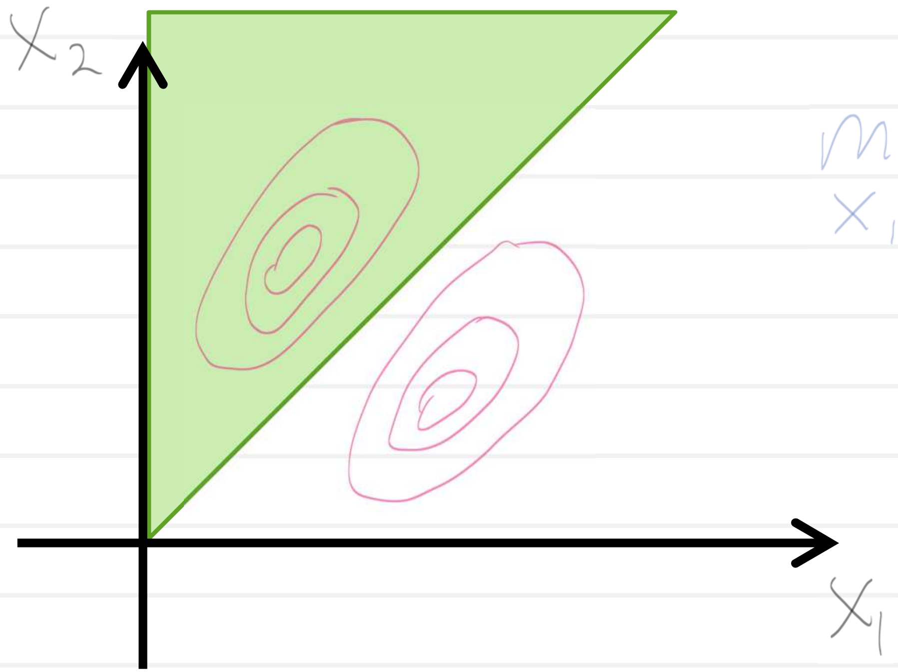
Opening two stops





$$\min_{x_1, x_2} u(x_1) + u(x_2) + \sum_s \min_{1,2} \Big(d(y_s, x_i) + d(x_i, Sk) \Big)$$

Opening two stops



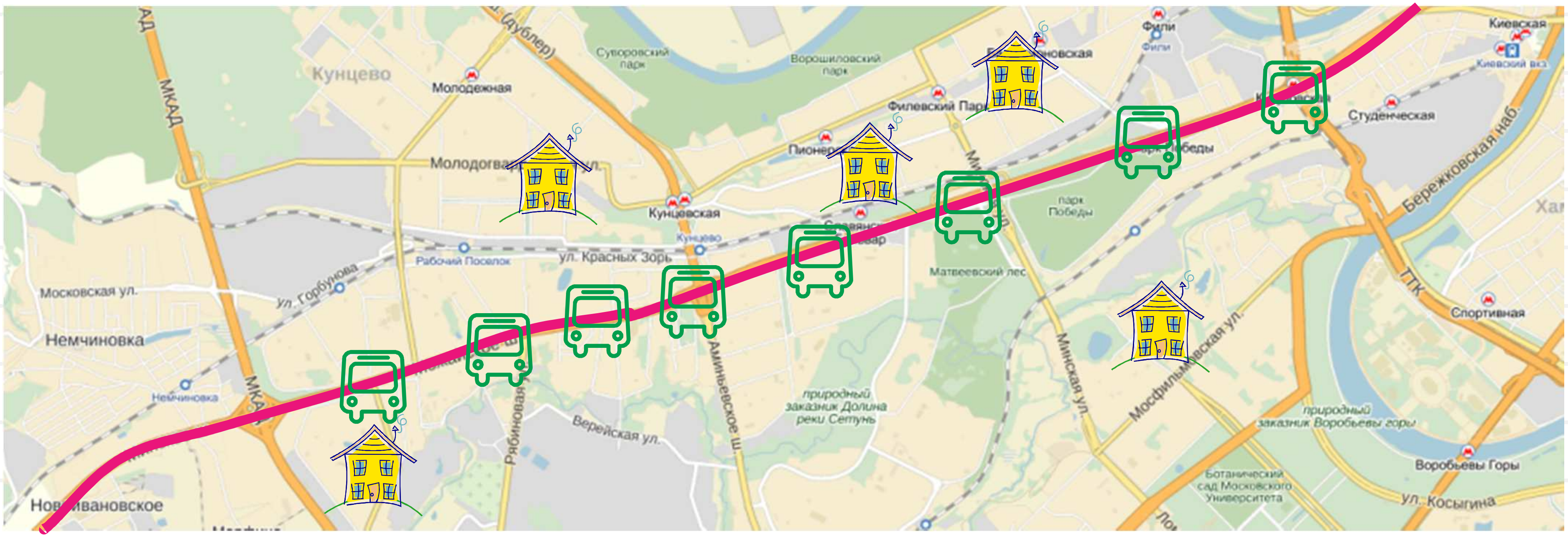
$$\min_{x_1,x_2} u(x_1)+u(x_2)+\sum_s \min_{1,2}\big(d(y_s,x_i)+d(x_i,S_k)\big)$$

Getting on the further of the two stops incurs further time penalty. How to model this?

$$\min_{x_1\,x_2}\Big(u(x_1)+u(x_2)+\sum_s \min\big(d(y_s,x_1)+d(x_1,S_k)+T;\ d(y_s,x_2)+d(x_2,S_k)\big)\Big)$$

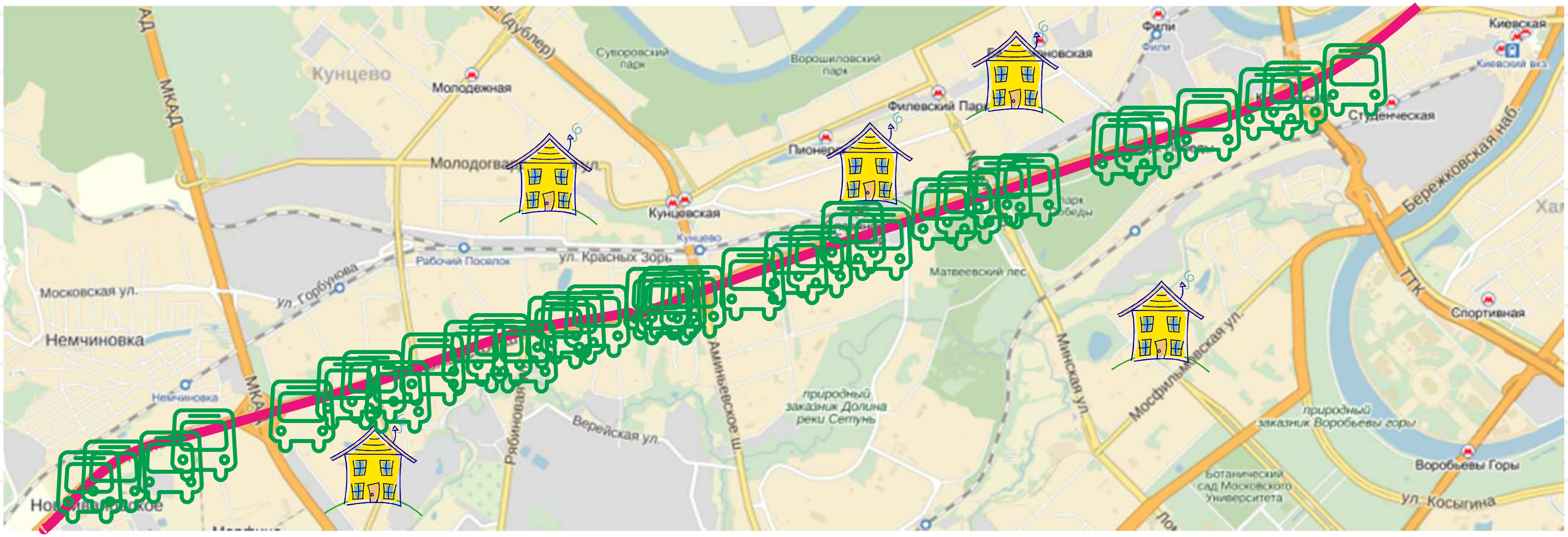
$$\text{s.t.:}\quad x_1 \le x_2,\quad x_1\in[0;20],\quad x_2\in[0;20].$$

What about more stops?



- From a certain number, exhaustive search becomes infeasible
- What about making the number of stops part of optimization?

Different modeling approach



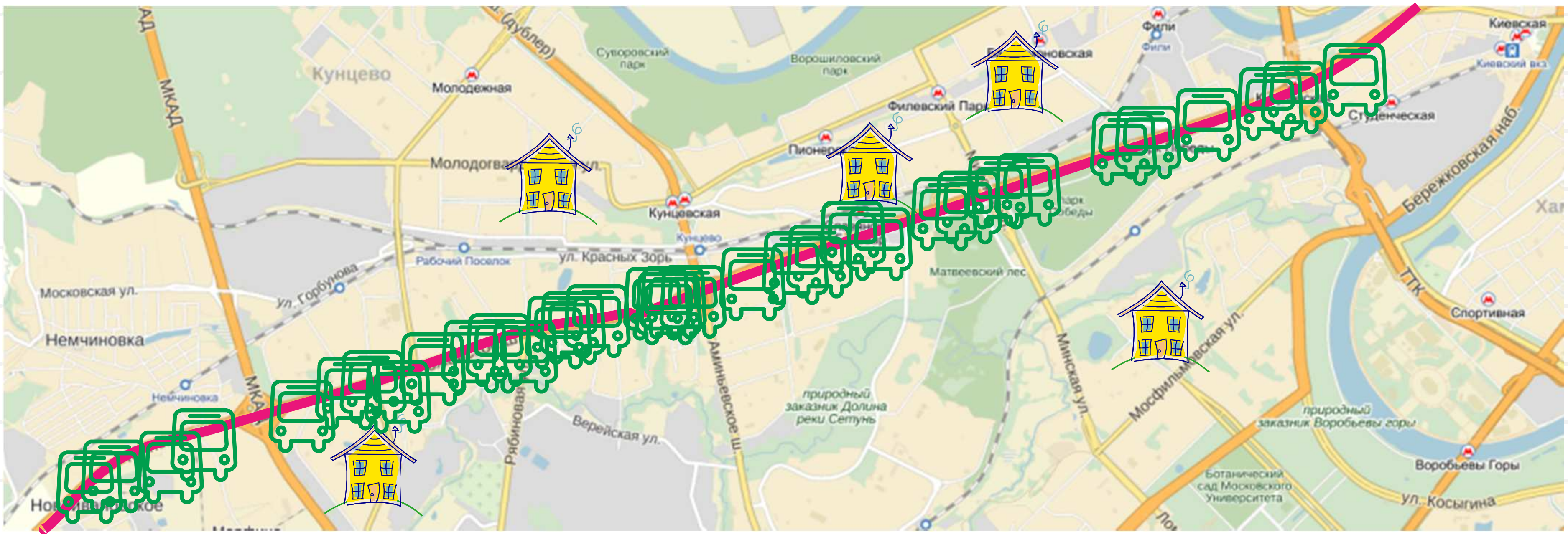
$x_i, \ i = 1 \ldots N$ fixed candidate locations of the stops

$y_s, \ s = 1 \ldots M$ fixed locations of the students

$z_i, \ i = 1 \ldots N$ should we open the stop?

$h_s, \ s = 1 \ldots M$ which stop should the student use?

Different modeling approach



$$\min_{z,n} \sum_{s=1}^{M} \left(d(x_{n_s}, y_s) + d(x_{n_s}, S_k) \right) + \sum_{i=1}^{N} u_i z_i$$

$$\text{s.t.:} \quad \forall s \quad z_{n_s} = 1$$

$$\forall i \quad z_i \in \{0;1\} \qquad \forall s \quad n_s \in \{1,2\ldots N\}$$

An instance of the *facility location* problem

What do we want from the modeling step?

1. The objective function should reflect the task we are facing (not at all trivial!)

2. The domain must exclude unacceptable solutions (especially those with the low objective) and include all acceptable solutions (especially those with low objective)

3. The resulting formulation (the *program*) should be amenable for the efficient optimization

Quite often, we have to seek trade-off between 1+2 and 3

Evaluating the function

Main assumption:

- We can evaluate
- We can check if a point is within the domain

$$\min f_0(x)$$
$$\text{s.t.: } x \in D$$

Common additional assumptions:

- We can evaluate first-order derivatives
- We can evaluate second-order derivatives

$$\min f_0(x)$$
$$\text{s.t.:}$$
$$f_i(x) \le 0$$
$$h_i(x) = 0$$

What do we want from an optimization algorithm?

- Ideally, we want all **global minima**
- Well, maybe at least one

$$\boxed{\begin{aligned} &\min f_0(x) \\ &\text{s.t.: } x \in D \end{aligned}}$$

Examples of functions with multiple global minima

Definition: $\tilde{x}$ is a global minimum, if

$$\forall x \in D \qquad f_0(x) \geq f_0(\tilde{x})$$

Local minima

$$\min f_0(x)$$
$$s.t.: x \in D$$

- In many cases, finding a global minimum is not possible.
- Typically, we should still be able to find a **local minimum**

Definition: $\tilde{x}$ is a local minimum, if

$$\exists N(\tilde{x}) \subset D, \quad \forall x \in N(\tilde{x}) \quad f_0(x) \geq f_0(\tilde{x})$$

(D must be a topological space)

- Sometimes (non-smooth, high dimensional functions, complex domains), even getting to a local minimum is hard

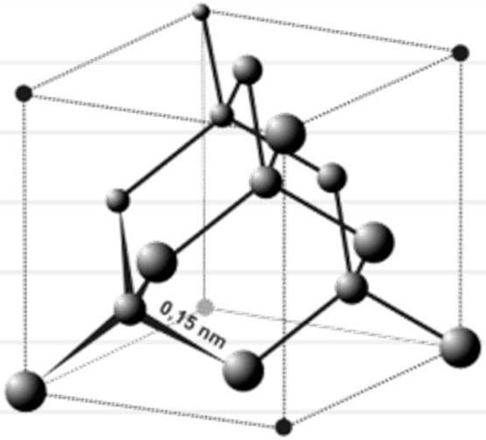
Topological space (from Wikipedia)

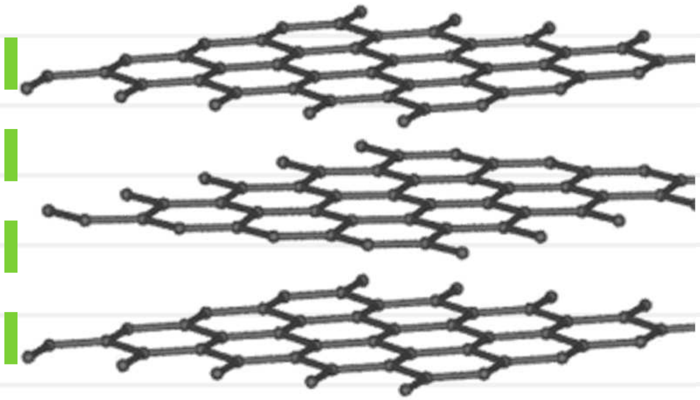
Let X be a set; the elements of X are usually called *points*, though they can be any mathematical object. Let **N** be a function assigning to each x (point) in X a non-empty collection **N**(x) of subsets of X. The elements of **N**(x) will be called *neighbourhoods* of x with respect to **N** (or, simply, *neighbourhoods of* x). The function **N** is called a neighbourhood topology if the axioms below are satisfied; and then X with **N** is called a **topological space**.

1. If N is a neighbourhood of x (i.e., $N \in$ **N**(x)), then $x \in N$. In other words, each point belongs to every one of its neighbourhoods.
2. If N is a subset of X and contains a neighbourhood of x, then N is a neighbourhood of x. I.e., every superset of a neighbourhood of a point x in X is again a neighbourhood of x.
3. The intersection of two neighbourhoods of x is a neighbourhood of x.
4. Any neighbourhood N of x contains a neighbourhood M of x such that N is a neighbourhood of each point of M.

Local minima in nature

Image from Wikipedia





Local minimum | Global minimum

How to check that we are at local minimum?

Assume:

- There are no constraints
- The domain is continuous
- The objective is smooth

Taylor expansion:

$$f(x) \approx f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x})$$

$$f(x) \approx f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x}) + \frac{1}{2} (x - \tilde{x})^T \nabla^2 f(\tilde{x}) (x - \tilde{x})$$

How to check that we are at local minimum?

Taylor expansion:

$$f(x) \approx f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x})$$

Necessary condition (for $\tilde{x}$ to be a local minimum):

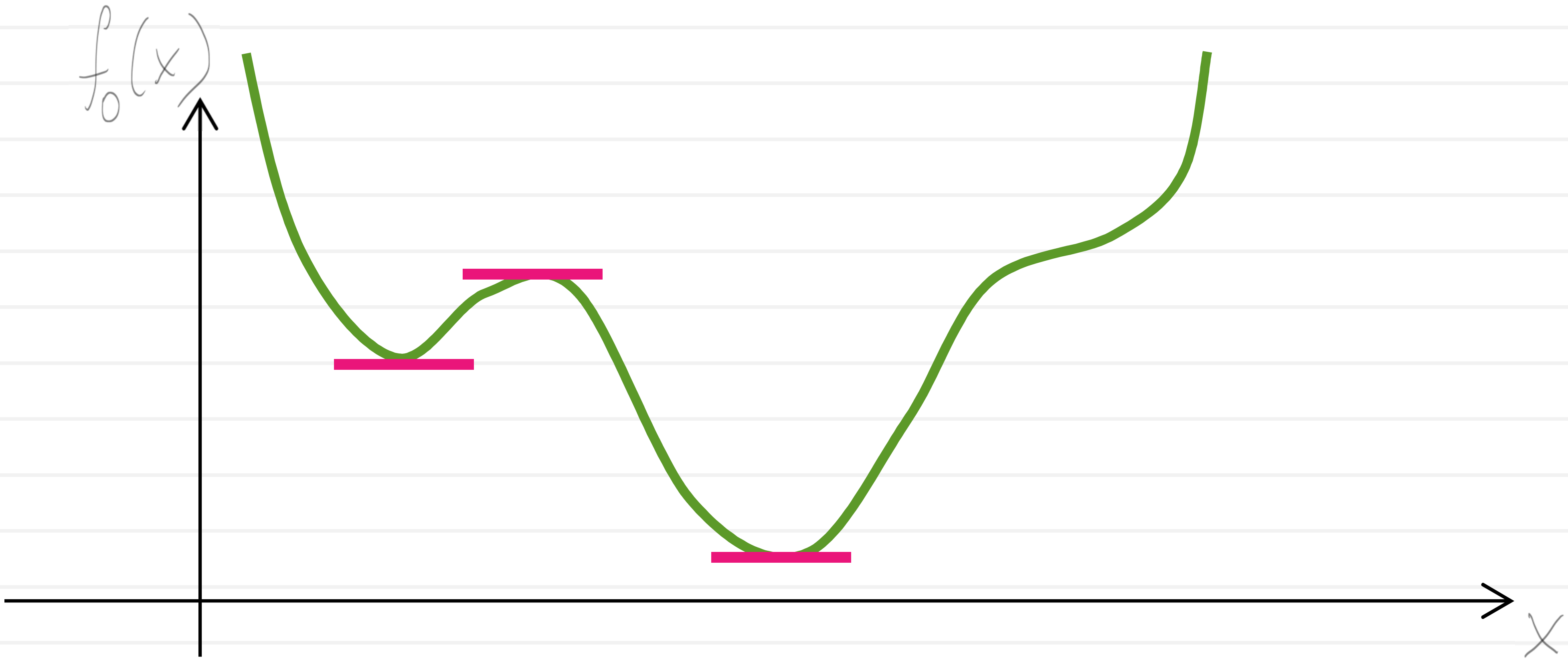
$$\nabla f(\tilde{x}) = 0$$

Proof (idea): assume converse, let $x = \tilde{x} - \alpha \nabla f(\tilde{x})$

$$f(x) \approx f(\tilde{x}) - \alpha \nabla f(\tilde{x})^T \nabla f(\tilde{x}) \approx f(\tilde{x}) - \alpha \|\nabla f(\tilde{x})\|^2$$

Thus, for small enough α $\quad f(x) < f(\tilde{x})$

1D example



Gradient descent

(Our first optimization algorithm)

$$\min f_0(x)$$

$x = x_0$

$\gamma = 0.001$

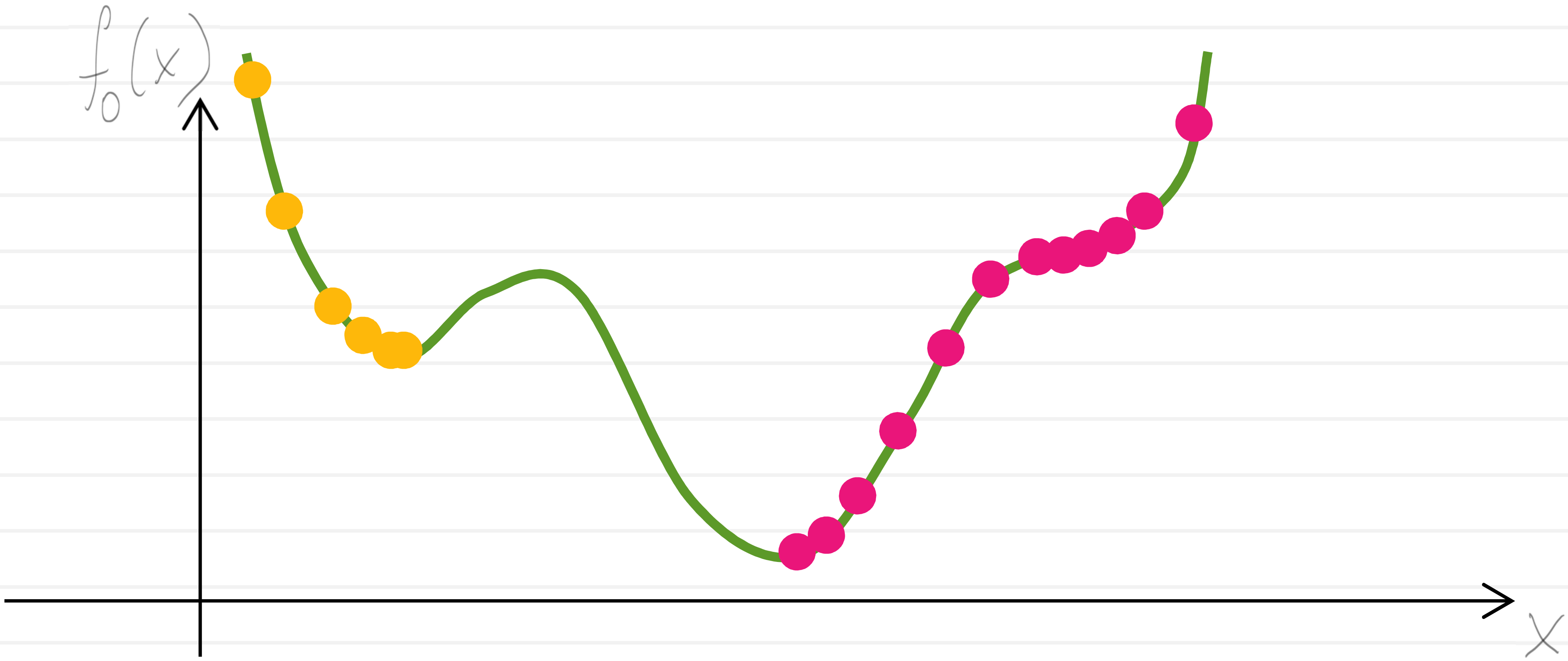
while $\|\nabla f_0(x)\| > \varepsilon$

$\quad x := x - \gamma \nabla f_0(x);$

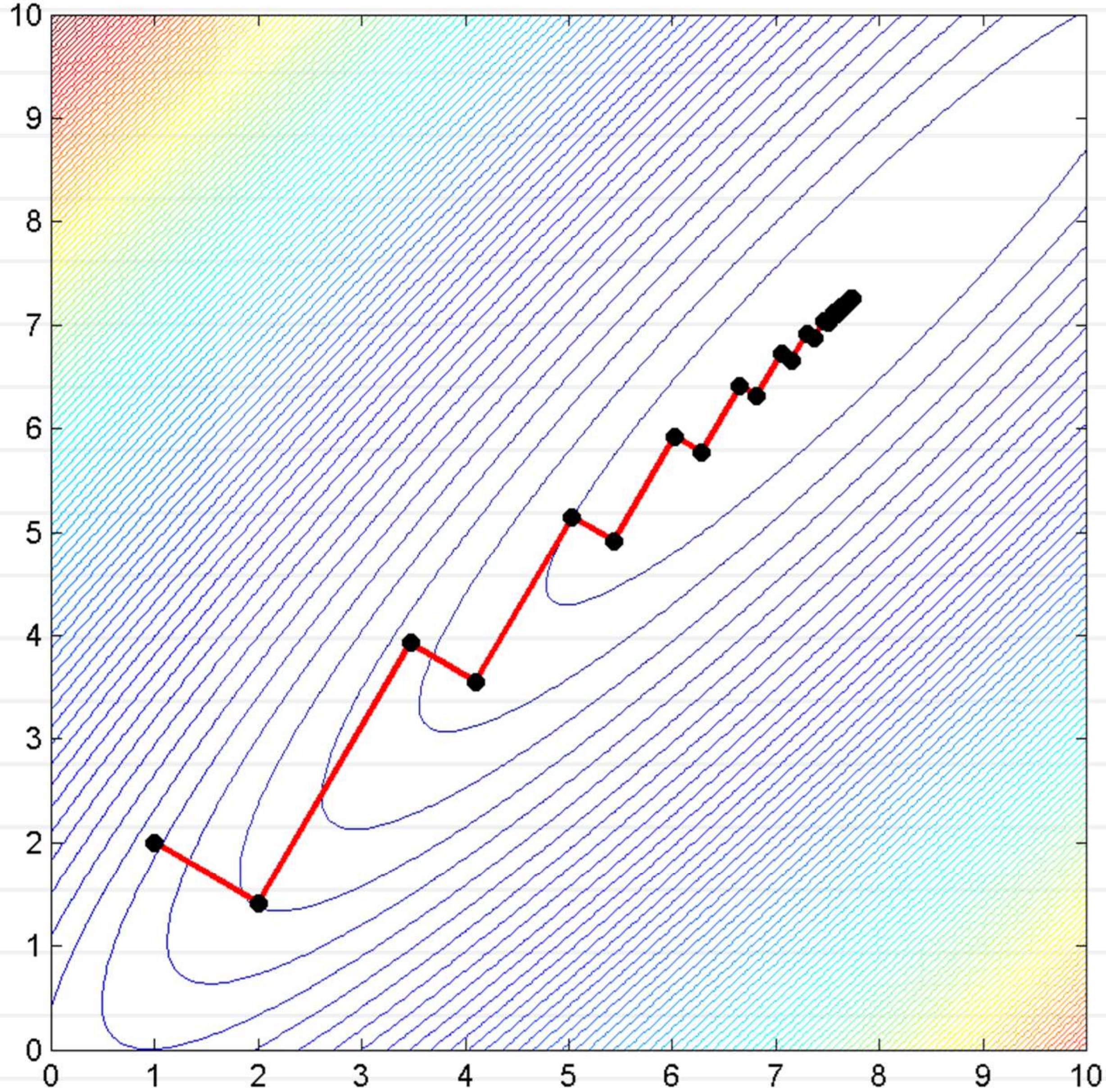
In practice, a care should be taken to adjust γ

More on unconstrained optimization in week 4

1D example



2D example

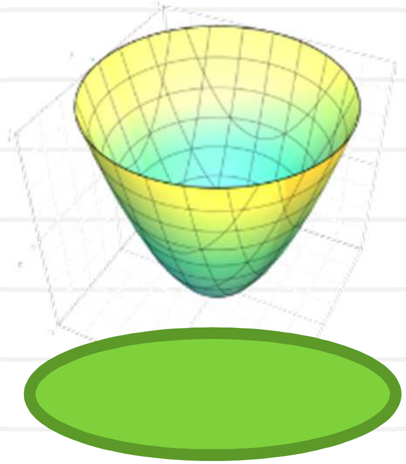


Checking for globality

Assume, we are at a local minimum.

How do we know, it is global?

- Generally, this is not possible, unless the problem has some special structure
- Most important case, when we know that we are at global minimum, is when the domain and the objective are *convex*



More on convexity later in Week 5

Minimization vs. Maximization

$$\boxed{\begin{array}{l} \max f_0(x) \\ \text{s.t.: } x \in D \end{array}} \quad \Longleftrightarrow \quad \boxed{\begin{array}{l} \min -f_0(x) \\ \text{s.t.: } x \in D \end{array}}$$

- Maximization is trivially reduced to minimization
- We will tend to discuss minimization, but it will all be applicable to maximization

Course schedule

Week 1	Local opt, DynProg	Python
Week 2	LP intro, Simplex method	CVXPY
Week 3	Networks, Integer programs	CVXPY+ GurobiPY
Week 4	Least sq., Unconstrained opt	SciPy. optimize
Week 5	Convex opt, duality	CVXPY
Week 6	Advanced	Misc
Week 7	Project work	
Week 8		

Assignments

- 6 assignments, 1 per week, 70 points total
- Assignments are due Wednesday, 23:59 of the next week
- The deadline is hard, after deadline you get 50% of points
- Submission: JohnDoe3.zip (IPython notebook *or* pdf + MATLAB code)

Collaboration policy for assignments

- You are welcome to discuss lectures and general concepts
- Assignments are for **personal** work
- Do not discuss a problem until you have thought about it for at least 30 minutes
- Your code **must** be written by you
- Copying code is **not** allowed
- It is ok to seek help with Python
- …but the code and the report you submit **must** be yours

Coding

Main environment: IPython

- +CVXPY (http://www.cvxpy.org/)
- +SciPy.optimize
- +GurobiPY (http://www.gurobi.com/)

Alternative environment: MATLAB

- +CVX (http://cvxr.com/cvx/)
- +MATLAB optimization toolbox
- +GUROBI

Lectures

- 18 lectures, including 1 guest lecture (Panagiotis Karras) on Nov, 6th
- Interactive (short exercises, discussions welcome)
- 2-2.5 hours
- PPTX (+voiced screencast) will appear on Stellar the night after the lecture
- Attendance recommended, not mandatory
- **No electronic devices please**

Getting help from the team

- Please use PIAZZA as much as possible! We will strive to respond quickly
- Offline recitations are once a week (as scheduled). Walk-in for consultation
- I will answer questions during and after lectures
- Other offline consultations are by email appointment with TAs

Self-study

- Some lectures (especially towards the second half) will follow textbooks closely

- Links to textbooks and MOOCs will be provided during the lectures

- Wikipedia is often quite useful

Projects

- Group projects (ideal size 3-4)
- Outcomes:
  - presentation
  - code
  - report
- upto 30 points for each participant
- You can either suggest your own topic or get one from us
- Topic and groups must be selected by the end of week 5
- Try to coordinate with the other class (joint project is ideal)

Problem: assign students to teams to projects

- Optimize assigning students to teams, and teams to projects

Things to consider:

- Preferences
- Each team should have enough coding experience close to average over all students
- Team size 3 or 4
- 0-2 teams per project

Write it as an optimization program
All your variables must be binary!