A Two-Staged Optimization Model for Generating a Facility Layout Using the Gradient Descent Approach

David den Ouden
TU Delft
Delft, The Netherlands
d.denouden-1@student.tudelft.nl

First mentor - assistant Professor Dr. Ir. P. Nourian
Second mentor - assistant Professor Dr. Ir. P.W. Heijnen
Board of examiners delegate - Geert Coumans

***Abstract*—In this paper a two-stage model is proposed, combining the idea of a two-stage model from Anjos and Vieira [2016] with a gradient descent approach, much like proposed in Sikaroudi and Shahanaghi [2016], in order to solve the facility layout problem for problems consisting of 8 and 12 departments . The gradient descent approach uses the partial derivatives of a multi-variable objective, the flow-cost, in order to get a vector which is the direction of greatest descent. Computing this vector for all departments and then moving them creates an iterative improving loop. In addition to the gradient descent approach a swapping procedure and a shooting procedure is introduced in order to reduce the effect of random starting position.The proposed method was tested on two toy-problems from Tam and Li [1991] and compared to results existing in literature. Additionally, the effect of including a first-stage model was tested as well. When looking directly at the results, the proposed method is consistent and shows good results. Especially the inclusion of the first-stage model helps finding better solutions. Compared to other results found in the literature, the method is slightly lacking based purely on the objective value.**

I. Introduction

In a world that is getting more complicated, so are the floor-plans of numerous buildings. Especially for industries, a layout of the floor-plan can really define how profitable a factory is, it is estimated that in total 20-50% of total operating expenses can be related to the material handling costs and the layout of a factory [Tompkins et al., 2010]. Furthermore, early changes in the design of a layout have a big impact on the final design and can be very profitable, especially considering that changes in a later stage are significantly harder, costlier and more complicated to perform. Chwif et al. [1998] suggest that the optimal location of facilities is one of the most important issues that should be resolved early in the design stage. In the past, when layout were less complicated, optimizing the location of facilities was not as much of an issue. However, after the second world war, operations research and with it the optimizing of processes arose. Optimizing various process has been a big issue ever since, and so too the optimization of the layout of facilities emerged which became known as the Facility Layout Problem. With the development of powerful computers that are available to the wide public, a window of opportunities opened to be able to solve these "FLP's".

The facility layout problem deals with finding the optimal relative locations of departments on a plane.The facility layout problem can essentially be taken apart into two different components, those are the formulation of the problem and the resolution of the formulated problem. Drira et al. [2007] gives a comprehensive survey of the facility layout problem and talks about both the formulation as the resolution of it. There are generally two ways to formulate the facility layout problem geometrically: discrete and continuous.

Quadratic layout formulation, or put short "QAP", which stands for the quadratic assignment problem, is a kind of formulation to the facility layout problem that is discrete in nature. QAP is used by many researchers [Chwif et al., 1998] and is proven to be NP-Complete [Sahni and Gonzalez, 1976]. This means that the global solution can be found, but the time required to do so increases rapidly as the number of components increases. With QAP formulations, the plant site is divided into different blocks of the same size and each facility will be assigned to one or more of these blocks (in the case of unequal areas) [Drira et al., 2007].

In contrast to the discrete formulation of the problem, the facility layout problem is often formulated as continual. This continual formulation is often abbreviated to mixed integer programming MIP [Das, 1993]. Continual means that unlike the QAP, facilities can be placed anywhere and the main constraint is that they must not overlap, not even partially [Das, 1993] [Merrell et al., 2010] [Dunker et al., 2005]. The facilities are no longer bound to small block-sized elements, but are defined by their centroid coordinates (often: x_i, y_i) and their half length l_i and half width w_i [Drira et al., 2007]. Another way to formulate the positions of the facilities is by the coordinates of one of their corners (most often the bottom left) and their full width w_i and length l_i [Chwif et al., 1998].

A third way layouts are sometimes formulated is in the shape of a graph, using graph theory. In this formulation, the departments take the shape of vertexes and connections between the departments are represented by the edges between the vertexes, thus forming an undirected graph. However, to eventually get to a non-abstract layout, a graph theoretic approach must make use of either a MIP or QAP formulation

nonetheless.

When it comes to resolving the formulations, there are a number of different approaches, which can be divided into exact approaches and meta-heuristic approaches. Exact approaches aim to explore the entire solution space, and can therefor always find the global optimum, because all solutions have been evaluated. An example of such an approach is presented by Kouvelis and Kim [1992], who make use of a branch and bound algorithm. Another example is by Kim and Kim [1999], who also implement drop-off and pick-up points for the departments. However, like the numerous other articles that dealt with exact methods, these methods are often incapable of solving cases with a large number of departments (up to fifteen), making meta-heuristics much more feasible for larger sized problems [Drira et al., 2007].

Amongst the different meta-heuristic methods, the most popular ones are the simulated annealing approach, the tabu search approach and the genetic algorithm approach. The first two are considered to be global search methods while the latter is classified as an evolutionary method [Drira et al., 2007]. All three of these methods are however improvement algorithms, based on an initial solution and altering it slightly to iterate towards a more optimal solution. This is the complete opposite of construction algorithms, which, as the name might imply, construct a solution only once, carefully evaluating all steps before taking one. In practise, these two type of algorithms are often combined, as construction algorithms can provide a healthy starting solution for an improvement algorithm [Drira et al., 2007].

The method that is proposed in this paper consists of two models, the first model uses the gradient decent approach where the departments will be represented as circles in an attempt to find **the optimal relative location constraints** to be used in the second model, which will be a linear constraint optimization solver. This method is further explained in section three, after the problem has been defined in section two. Section four takes the method into practise by applying it to existing toy problems, as well as a comparison with a real-life factory. Section five concludes this paper.

II. Problem Formulation

In this thesis, the facility layout problem is defined as a continuous problem, where the facilities can be placed anywhere on the map and can not overlap. The voxelated approach of the quadratic assignment problem leaves the problem that the definite shape of one department (unless it consists solely of one voxel) is hard to influence. For example, an aspect ratio in the discreet environment would ultimately be very hard to implement using the quadratic assignment problem. Scholz et al. [2009] describes that the main drawback on discreet approaches and graph theoretic approaches is the geometric constraints that can not be considered sufficiently. In the case of this problem, the geometric constraints will prove to be very important.

A. Shaping the departments

The method is primarily based on the assumption that it should be feasible for producing layouts for factories, in particular, vegetable food factories. Since in these type of factories the departments are split heavily between departments that can take multiple dimensions (e.g. a storage room with a flexible organisation of storage racks) and constraint departments that can really only take on one set of width and height (e.g. a room with one big machine). Hence all departments are defined by their area (A_i) and max aspect ratio a_i [Anjos and Vieira, 2016]:

$$max\left\{\frac{w_i}{h_i}, \frac{h_i}{w_i}\right\} \leqslant a_i \tag{1}$$

$$w_i h_i = A_i \tag{2}$$

The aspect ratio constraints will be discarded in the first-stage model (graph decent) as its purpose is to find relative relations between departments, hence, all departments in the first-stage model will be modeled as circles with radius r_i and a set of coordinates x_i & y_i. In the second-stage model, all departments will be modelled as squares by assigning a width w_i and height h_i and a set of coordinates x_i & y_i.

B. Overlap

In order to restrict the departments from overlapping with the total facility two constraints are introduced in both the first-stage and the second-stage model:

$$x_i + \frac{1}{2}w_i \leqslant \frac{1}{2}W_F, \ and \ \frac{1}{2}w_i - x_i \leqslant W_F \tag{3}$$

$$y_i + \frac{1}{2}h_i \leqslant \frac{1}{2}H_F, \ and \ \frac{1}{2}h_i - y_i \leqslant H_F \tag{4}$$

Where W_F and H_F are equal to the total facility's width and height, respectively. In case of the first-stage model: w_i = h_i = r_i. The way that the overlap between facilities is constrained differs between the two models. In the first-stage model (gradient descent), overlap is allowed, however, it does add to the objective by adding the one dimensional overlap $r_i + r_j - d_{ij}$ to the distance in the case there is overlap. In this case, d_{ij} is the euclidean distance between the centroids of the departments.

$$d_{overlap} = \lambda_d(r_i + r_j - d_{ij}) \tag{5}$$

$$\lambda_d = \begin{cases} 0 & for\ r_i + r_j - d_{ij} \leqslant 0 \\ 1 & otherwise \end{cases} \tag{6}$$

In the case of the second-stage model, where the actual layout is computed, overlap is not allowed, and thus constraints are added to prevent the departments from overlapping. Since the solver does not take non-linear constraints, a boolean variable is added which is implemented in such a way that the absolute value between coordinates is found, see equations (7) through (9).

$$\lambda_{ij} = \begin{cases} -1 & for \;\; x_i \leqslant x_j \; and \; y_i \leqslant y_j \\ +1 & otherwise \end{cases} \tag{7}$$

$$2\lambda_{ij}(x_i - x_j) \geqslant w_i + w_j \tag{8}$$

$$2\lambda_{ij}(y_i - y_j) \geqslant h_i + h_j \tag{9}$$

C. Objective

The objective of this method is to minimize the flow-cost of the layout. For a two department layout, this would be the flow between the departments, (f_{ij}), multiplied by the distance between the departments, (d_{ij}), multiplied by the cost per flow per distance of moving materials between the two departments, (c_{ij}). Naturally, the flow-cost for a layout consisting of more than two departments would then be the sum of the flow-costs for all sets of departments. This then leads to the following objective for the flow-cost, which is similar to the objective for minimizing the material handling costs as found in Wang et al. [2005].

$$MinFlowCost = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} f_{ij} d_{ij} \tag{10}$$

The flow f_{ij} between the departments is provided using a matrix of i rows by j columns, providing the flow between departments i and j at row i and column j. The cost c_{ij} is taken to be one for simplification. The distance d_{ij} between the departments is different for both models. In the first-stage model, the distance is measured as the euclidean distance as seen in equation (11), additionally, the distance for overlapping, as seen in equation (5) is assimilated in the distance equation . The second-stage model however, will make use of the rectilinear distance, like seen in equation (12), as the chosen solver is only capable of performing linear operations.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + d_o verlap \tag{11}$$

$$d_{ij} = |x_i - x_j| + |y_i - y_j| \tag{12}$$

D. Relative location constraints

The whole idea of splitting the method into two models is to gain an early insight in what arrangement of departments would work near optimal without requiring too much computational power. This early insight does not have to be accurate in aspect ratio or other geometric constraints, nor in overlapping. The loss off accuracy in that case gains the benefit that the model becomes much faster. This way, a near optimal arrangements can be found in the first-stage model and this arrangements data can be passed onto the second-stage model, where all previous requirements must be met, but due to the extra constraints found in the first-stage model the second-stage model should be able to converge faster.

The second-stage models therefor takes all constraints that were found in the first-stage model. These will be constraints between all departments i and j that will separate them either horizontally or vertically. To determine this separation the differences between x and y coordinates of both departments is evaluated. Should δx be 1.5 times bigger than δy, then a separation constraint will be issued to separate department i to the left of j if i was to the left of j in the most optimal solution from the first-stage model, or to the right of j if i was to the right of j in the most optimal solution from the first-stage model. In the same way, should δy be 1.5 times bigger than δx, then a separation constraint will be issued to separate department i to the top of j if i was to the top of j in the most optimal solution from the first-stage model, or to the bottom of j if i was to the bottom of j in the most optimal solution from the first-stage model. If neither of the two differences is significantly bigger than the other (a factor 1.5), no constraint will be issued. The idea to split the method into two models to first find these relative location constraints and how to find these constraints was originally proposed in Anjos and Vieira [2016]. The main difference is that in that paper, the relative location constraints will be issued even if one δ is only slightly bigger than the other.

III. Proposed Methodology

The method proposed in this model is separated into two stages, a first-stage model and a second-stage model. The first-stage model is a simplified version that does not take into account aspect ratio and other geometric constraints and takes the overlap as a soft constraint. This is in order to quickly find a good solution and then transfer the relative location information to the second-stage model in the shape of constraints. This way, the hypothesis is that the second-stage model, which does take all the hard constraints as described in section 2 into account, will be able to converge into a good solution faster.

A. First-stage model

The first stage model makes use of the gradient descent approach to iteratively move towards a better objective function (10). This idea was already proposed in Sikaroudi and Shahanaghi [2016], where the term force exertion heuristics was used in order to get to a layout. The main difference with this approach is that here the gradient descent approach does not need to find a feasible solution, only an idea about the relative positions of the departments in a good solution. This gradient descent approach builds upon the fact that the gradient of any function can be used to find the direction of steepest ascent (or descent when taking the negative of this direction). This approach works for any objective and for multiple variables by taking all the partial derivatives and combining these in a vector, which is the direction of steepest ascent. This principle is explained in equation (13) for an objective f consisting of two variables: x and y.

$$\nabla f = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \tag{13}$$

$$f(x_i, y_i) = \sum_{j=1}^{n} f_{ij}(d_{ij} + \lambda_d(r_i + r_j - d_{ij})) \tag{14}$$

$$\nabla f(x_i, y_i) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad for\ \lambda_d = 1 \tag{15}$$

$$\nabla f(x_i, y_i) = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \quad for\ \lambda_d = 0 \tag{16}$$

$$\frac{df}{dx_i} = \sum_{j=1}^{n} \frac{(x_i - x_j) f_{ij}}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{17}$$

$$\frac{df}{dy_i} = \sum_{j=1}^{n} \frac{(y_i - y_j) f_{ij}}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{18}$$

The principle of the gradient descent is used in this first stage model by iteratively moving every department in the direction of greatest descent by taking the gradient of the objective. In this objective, the variables are the x_i and y_i coordinates of department i whilst the coordinates of every other department j are considered to be a constant. This gradient is then calculated for every department, after which the departments are moved resulting then in a different gradient thus creating an iterative process in which all departments move towards the lowest objective. This iterative process is explained in algorithm 3.

Interestingly, the partial derivatives between any two departments that are not overlapping ($\lambda_d = 0$) are equal to the difference in x-coordinates or y-coordinates divided by the distance between the two departments and multiplied by the flow, as seen in equations (17) and (18). In essence this would mean that the gradient would be a normalised vector that origins from one department and points to the other. When $\lambda_d = 1$, however, all the variables disappear from the objective, leaving the gradient to be [0,0], meaning that any direction and no direction will improve the objective considering just these two departments.

An important addition is the inclusion of the overlap repulsion vectors, these vectors are introduced to remove overlap whenever there is overlap. The vector acts in the opposite direction of the department j that department i has overlap with. A last vector prevents the departments from overlapping with the total facility, although in this first-stage model the total facility is merely present to prevent the departments from moving out of bounds.

1) Shooting: In addition to the gradient descent approach, two procedures are proposed to solve certain problems that a gradient descent approach encounters. The first problem is that the the departments are seeking equilibrium, but sometimes this is a local minimum equilibrium. The downside of the equilibrium is that there is no natural way for the gradient descent approach to escape this local equilibrium once trapped. In order to counter this, the shooting procedure is introduced. This procedure initiates when there has been no progress in the objective for n iterations, after which the next iterations multiplies the flow attraction vector by a factor α. During this shooting iteration, all overlap vectors are ignored.

This procedure introduces an imbalance in the equilibrium so that the local minimum can be escaped. Once the shooting procedure has initiated, a second one can not be initiated for n iterations. Furthermore, during this cool-down period, the objective is stored but the solution does not qualify as valid until the cool down is reduced to 0 again. The procedure for shooting is summarized in algorithm 1.

2) Swapping: The second problem that occurs is the influence of the random starting positions. While the shooting procedure will already partly eliminate this problem, a swap procedure is also proposed. When triggered, the swap procedure evaluates all possible department swaps and ranks them based on their influence on the objective. It is important to notice that the overlap is not included in the objective when evaluating a swap, since this would further encourage swapping. When all possible swaps have been evaluated, the most profitable one (if any) is performed. The trigger for swapping is a rapid increase in the objective, where the current iterations must have improved by at least β % in comparison to objective of the previous iteration. Just like with shooting a cool-down of n iterations is issued where neither a swapping of shooting procedure maybe initiated.The procedure is summarized in algorithm 2.

B. Second-stage model

In the second-stage model, the relative location constraints from the first-stage model will be implemented in addition to all hard constraints as mentioned in section 2. The second-stage model takes all the constraints and objective and models that in Google's OR-tools [Perron and Furnon], using its CP-Sat solver, a linear constraint programming solver capable of running optimizations by using only integers. Since the solver is linear, some constraints involving divisions and absolute values had to be rewritten as mentioned in section 2. To allow for the implementation of the absolute values, the .EnforceOnlyIf() method provided by the CP-Sat solver was used. A non-linear constraint that cannot be rewritten is the area requirement as seen in equation 2, it is non-linear as it multiplies two variables.. The CP-SAT solver offers a way that still allows this multiplication of different variables through the .AddMultiplicationEquality() method. Together with the fact that only integers are allowed, and the constraint can only be set to equality (no bigger then or less than), the number of options that are left to shape a department are low, since the area requirement only has so many ways to be a product of two integers. A possible way to work around this is the up-scaling of the variables and the area requirements, every factor 10 will add an extra decimal to the options of forming the area requirement. A nice example of this up-scaling is shown in table I.

Algorithm 1 Pseudo-code for the "shooting" procedure

```
if cool-down != 0 then
    cool-down = cool-down -1
end
if iteration > n AND cool-down == 0 then
    if objective_current >= objective_iteration-n then
        shoot = True
    end
end
if shoot == True then
    Compute only the flow-attraction vectors
    Move all departments α * the set distance
    cool-down = n
    shooting = False
end
else
    Move departments normally
end
```

Algorithm 2 Pseudo-code for the "swapping" procedure

```
if cool-down == 0 then
    if objective_current/objective_previous ≤ β then
        swapping = True
    end
end
if swapping == True then
    flag = False for i in range(nr_departments) do
        for j in range(nr_departments) do
            if i !j then
                copy the data frame
                swap coordinates of department i and j
                calculate the objective excluding overlap if
                objective < min(objectives) then
                    save i
                    save j
                    flag = True
                end
            end
        end
    end
    if flag == True then
        swap departments i and j in actual data frame
        cool-down = n
    end
    swapping = False
end
```

Area Requirement	Nr of possible integer multiplications
12	3
1200	15
120000	35
12000000	63
56	4
5600	18
560000	40
56000000	70

TABLE I
THE AMOUNT OF INTEGER MULTIPLICATIONS TO MAKE THE AREA REQUIREMENT, UP-SCALING BY FACTORS OF 100 OFFERS MORE POSSIBILITIES, BUT THIS COMES AT A COST OF COMPUTATIONAL POWER REQUIRED AS THE VARIABLES WILL BE SCALED AS WELL (BY FACTORS 10).

Algorithm 3 Pseudo-code for the first-stage model

```
Import necessary modules
Import data from excel using pandas
for i in range(nr_departments) do
    Radius = sqrt(Area/π)
    x = random_integer (min,max)
    y = random_integer (min,max)
end
for k in range(max_iterations) do
    for j in range(nr_departments) do
        for j in range(nr_departments) do
            if i!=j then
                Compute partial derivative df/dx_i
                Compute partial derivative df/dy_i
                Compute flow attraction vectors
                Compute repulsive vectors for overlapping
                Compute repulsive vectors for overlap with
                outside facility
                Add weights to all vectors
                Compute the master vector for each department
            end
        end
    end
    Move the departments in the direction of the master-vector
    a set distance
    Compute the Objective
    for i in range(nr_departments) do
        for j in range(nr_departments) do
            if i!=j then
                Objective_ij = distance_ij * flow_ij
                Objective = Objective + Objective_ij
            end
        end
    end
    Append Objective to Objectives
    if Objective == min(Objectives) then
        Save x and y positions
        Save iteration number
        Save objective and overlap amount
    end
end
```

IV. Results

The proposed frame-work was tested using a python 3.8 environment set up in Rhinoceros [Associates, b] using the plugin grasshopper [Associates, a] for the first-stage model and the CP-Sat solver from google OR-tools [Perron and Furnon] modelled in python for the second-stage model. The tests were done using an Intel(R) Core(TM) i7-9700K CPU @3.6GHz with 32 Gb of memory. As test problems, two cases were taken from [Tam and Li, 1991] and [Chwif et al., 1998] which were originally composed in [Nugent et al., 1968] consisting of 8 and 12 departments. Both the results for the first- and second-stage model are presented.

A. *first-stage model*

The results for the first model are published in table II. The table holds the results that were taken for a 50 sample test on the four different methods, vanilla, swapping, shooting and swapping and shooting combined. Each method had its own sample of 50 runs, taking a maximum of 800 iterations.

Each column provided the results for one of these methods, going over the objective as well as the flow-cost, the amount of overlap and the number of iterations it took to find the best result. Each subsection is split into the average, minimum, maximum and standard deviation. Table II shows the results for the 8 department test, only the 8 department problem was used to test the different methods efficiency. Additionally, the relative improvement of the proposed improvement procedures compared to the vanilla method can be found in table III.

The computational time needed for the test problems were on average 20.1 seconds for the 8 department problem and on average 29.8 seconds for the 12 department problem for 800 iterations. Figures 1 and 4 show the best layout for the first-stage model for respectively the 8 and 12 department problems, as well as their graphs. The graphs show the improvement of the objective, flow-cost, and overlap through all 800 iterations. The green circle encircles the best iteration (corresponding with the layout).

When comparing the different methods in tables II and III, it can be noted that the results for the vanilla method are the least promising, with the highest average objective, overlap and high standard deviations, indicating a heavy influence of the random starting positions. This influence is slightly reduced for the shooting method, as well as the average objective. However, the real improvements can be found with the swapping method and the method that includes both swapping and shooting, greatly reducing the average objective and the standard deviation. While swapping has the best average objective, the method with both procedures has the absolute single best solution, which is due to the fact that it has a slightly higher standard deviation, increasing the chances to get a very low objective (and also a very high objective) while still, on average, producing very solid solutions. Therefor, it can be argued that the method that includes both procedures is preferred when there is enough time to run multiple tests while the swapping method would be preferred when only a few samples can be run.

	Swapping	**Shooting**	**Both**
Objective			
Average	-15%	-5%	-15%
Minimum	-2%	-2%	-2%
Maximum	-32%	-1%	-20%
Standard deviation	-83%	3%	-65%
Overlap			
Average	-70%	-36%	-58%
Minimum	35%	52%	33%
Maximum	-98%	-72%	-96%
Standard deviation	-96%	-72%	-94%
Flow-cost			
Average	-15%	-4%	-15%
Minimum	28%	30%	27%
Maximum	-33%	-2%	-20%
Standard deviation	-83%	-4%	-67%
Iterations			
Average	10%	29%	9%
Minimum	-	-	-
Maximum	0%	0%	-1%
Standard deviation	-17%	-24%	-29%

TABLE III

The improvement in percentages for the proposed improvement methods compared to the vanilla method for the 8 department problem, negative is a decrease (improvement).

	Vanilla	**Swapping**	**Shooting**	**Both**
Objective				
Average	905.8	766.4	864.7	767.6
Minimum	760.6	745.1	747.9	742.5
Maximum	1225.7	829.6	1207.9	977.9
Standard deviation	96.7	16.9	99.2	33.8
Overlap				
Average	4.9	1.47	3.14	2.04
Minimum	0.46	0.62	0.7	0.61
Maximum	146.41	3.34	40.7	5.73
Standard deviation	20.46	0.77	5.63	1.16
Flow-cost				
Average	894.1	757.3	854.6	755.8
Minimum	568	727.7	741	723.1
Maximum	1220.5	816.8	1200.8	971.4
Standard deviation	104.5	17.5	99.9	35
Iterations				
Average	475.8	521.9	613.1	518.8
Minimum	0	141	8	118
Maximum	799	799	796	790
Standard deviation	238.3	197.1	181.2	169.1

TABLE II

The results for the 8 department problem for the first-stage model.

B. *second-stage model*

In the second-stage model, the best solution found during the first-stage model for the 8 and 12 department problem, as seen in figures 1 and 3, are used to find the relational constraints. These constraints, together with all constraints established in section 2, will be modelled in the CP-Sat solver and solved for three time frames: 25 seconds, 100 second and 300 seconds. Additionally, the same models are solved but this time the relational constraints from the first model are excluded in order to test the effectiveness of the first model.

The results for the 8 department problem, taking into account 3 different time frames and the in- or exclusion of the first-stage model's constraints are published in table IV, while the same results for the 12 department problem are published

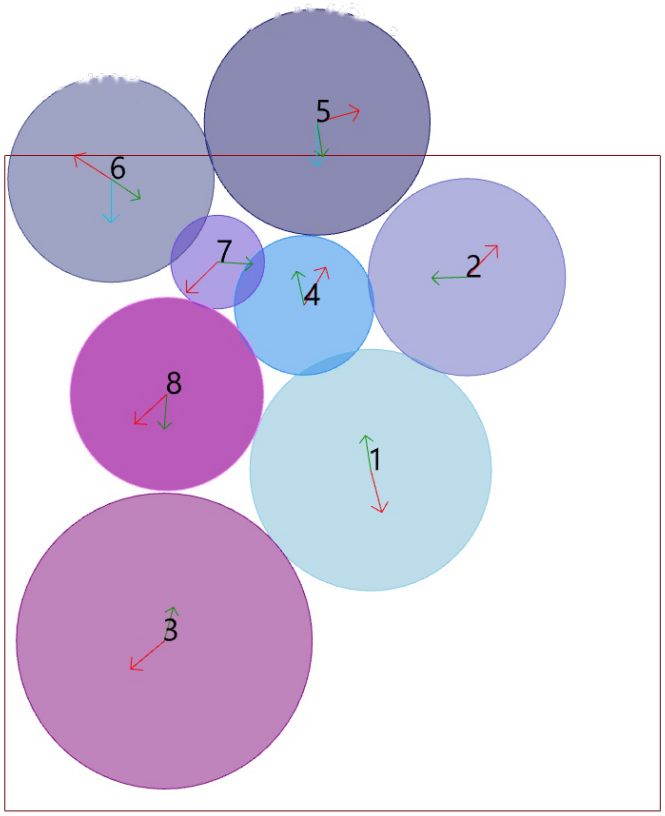

Fig. 1. The result for the first-stage model considering the layout for the 8 department problem. Source: Own work

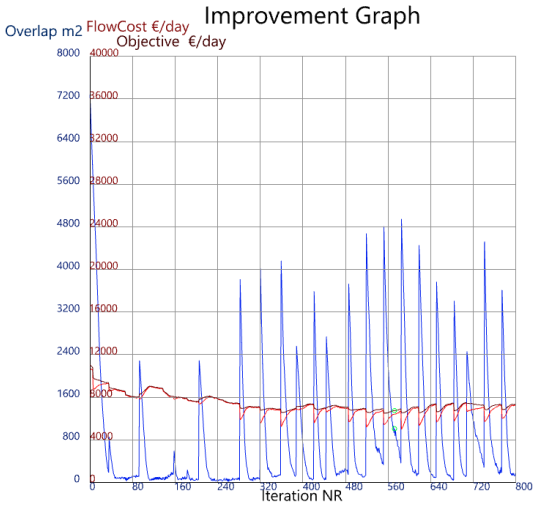

Fig. 2. The change of objective, flow-cost and overlap through the iterations concerning the best solution for the 8 department problem. Source: Own work

in table V. Additionally, the best results are published in table VI, which, in addition to providing information on the objective as stated in this paper, compares the objective

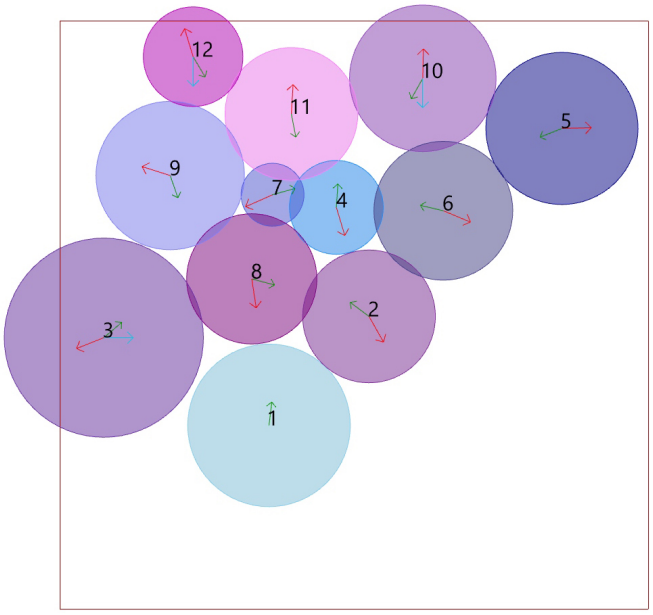

Fig. 3. The result for the first-stage model considering the layout for the 12 department problem. Source: Own work

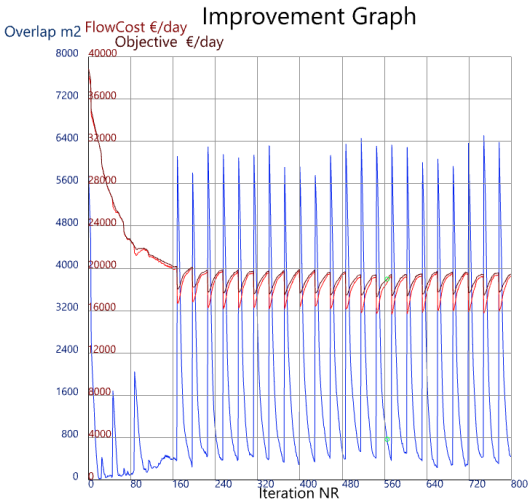

Fig. 4. The change of objective, flow-cost and overlap through the iterations concerning the best solution for the 12 department problem. Source: Own work

with up to two references who have solved the exact same problem. In order to be able to compare the objective, the final layout was found using the objective in this thesis, but an additional python component would calculate the objective for that solution as formulated in Tam and Li [1991], thus making the three methods comparable.

When comparing the inclusion of the first-stage model's constraints versus the exclusion of those constraints in tables IV and V , it can be said that the inclusion of the con-

	Relational constraints	No relational constraints	Difference%
25s	477.2	528.1	9.6
100s	427.6	520.3	17.8
300s	410.3	515	20.3

TABLE IV
THE RESULTS FOR THE 8 DEPARTMENT PROBLEM INCLUDING THE FIRST-STAGE MODEL'S CONSTRAINTS AND EXCLUDING THEM.

	Relational constraints	No relational constraints	Difference%
25s	1338.9	1678.4	20.2
100s	1302.1	1607.1	19.0
300s	1302.1	1444.0	9.8

TABLE V
THE RESULTS FOR THE 12 DEPARTMENT PROBLEM INCLUDING THE FIRST-STAGE MODEL'S CONSTRAINTS AND EXCLUDING THEM.

straints does help significantly to lower the objective as the improvement is always at least 9.6% in all time frames, as well for the 8 department problem as for the 12 department problem. When comparing the best results of the method with the results found in the literature however, it can be seen that the approach in this paper is still behind the best approaches found in the literature. As Chwif et al. [1998] already noted, the method used in Tam and Li [1991] is an exact method, so the results there are an absolute optimum, but that method becomes unfeasible after 15 methods, while the method in this paper does not. In addition to that, the empty space ratio (ESR) was better in this case, 20% versus 37% and 23%, proving that this method creates a more compact layout.

Departments	OV	Recomputed OV	Tam OV	Chwif OV
8	410.3	967.3	839	-
12	1302.1	3931.4	3162	3684

TABLE VI
THE TWO PROBLEMS COMPARED TO METHODS FROM CHWIF ET AL. [1998] AND TAM AND LI [1991]

V. CONCLUSION

In this paper a two-stage model was proposed, combining the idea of a two-stage model from Anjos and Vieira [2016] with a gradient descent approach much like proposed in Sikaroudi and Shahanaghi [2016]. The proposed method was tested on two toy-problems from Tam and Li [1991] and compared to results existing in literature. Additionally, the effect of including a first-stage model was tested as well.

When looking directly at the results, the proposed method is consistent and shows good results. Especially the inclusion of the first-stage model helps finding better solutions. Compared to other results found in the literature, the method is lacking based purely on the objective value. What it is capable of doing a little better however, is the reduction of the empty space ratio. Additionally, the proposed method is versatile enough and quite fast compared to other methods.

Would there have been space for additional research, other test problems of different sizes could have been explored more thoroughly. Additionally, a real-life example could have been used to test the proposed method on. An idea for generating even more promising lay-outs is experimenting with the way that the relational constraints are transferred from the first-stage model to the second-stage model.

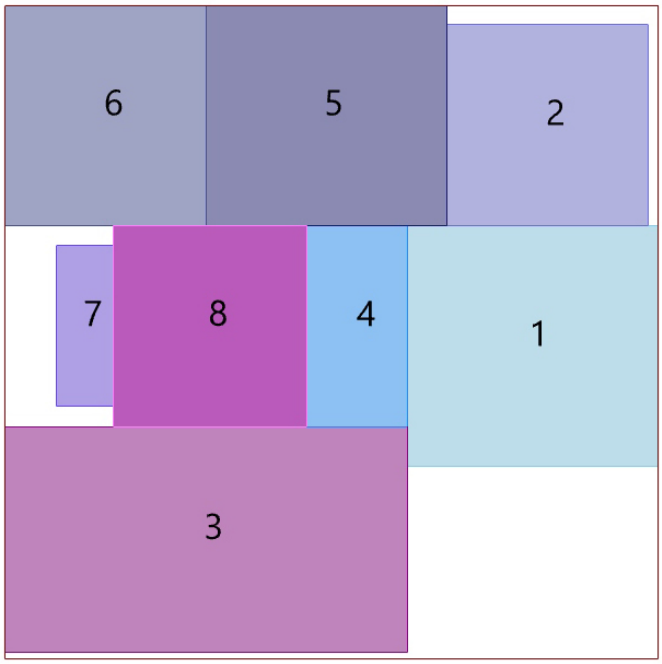

Fig. 5. The result for the first-stage model considering the layout for the 8 department problem. Source: Own work

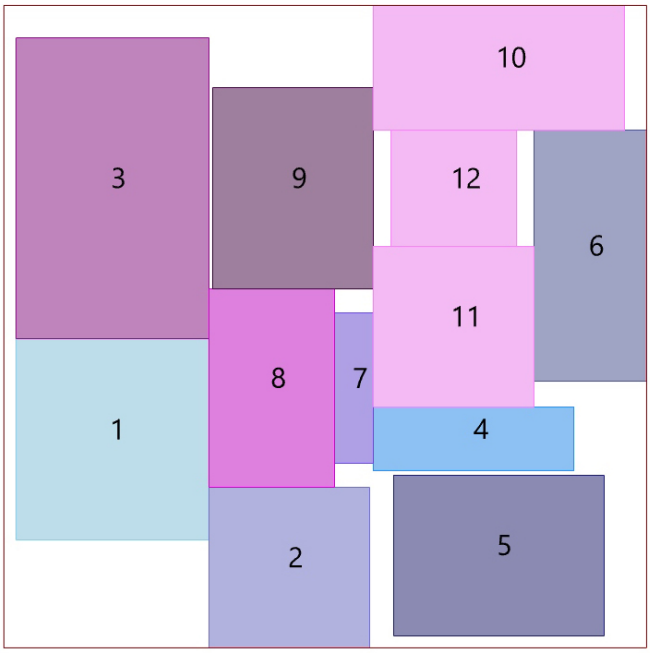

Fig. 6. The change of objective, flow-cost and overlap through the iterations concerning the best solution for the 12 department problem. Source: Own work

ACKNOWLEDGMENT

The author would like to express gratitude to his first and second mentors assistant Professor Dr. Ir. P. Nourian and ssistant Professor Dr. Ir. P.W. Heijnen from the TU-Delft for providing the guidance and feedback needed in order to write this paper.

REFERENCES

Miguel F. Anjos and Manuel V.C. Vieira. An improved two-stage optimization-based framework for unequal-areas facility layout. *Optimization Letters*, 10(7):1379–1392, 2016. ISSN 18624480. doi: 10.1007/s11590-016-1008-6.

Robert McNeel & Associates. Grashopper, a. URL https://www.rhino3d.com/6/new/grasshopper.

Robert McNeel & Associates. Rhinoceros 3d, b. URL https://www.rhino3d.com.

Leonardo Chwif, Marcos Ribeiro Pereira Barretto, and Lucas Antonio Moscato. A solution to the facility layout problem using simulated annealing. *Computers in Industry*, 36(1-2):125–132, 1998. ISSN 01663615. doi: 10.1016/S0166-3615(97)00106-1.

S. K. Das. A facility layout method for flexible manufacturing systems. *International Journal of Production Research*, 31(2):279–297, 1993. ISSN 1366588X. doi: 10.1080/00207549308956725.

Amine Drira, Henri Pierreval, and Sonia Hajri-Gabouj. Facility layout problems: A survey. *Annual Reviews in Control*, 31(2):255–267, 2007. ISSN 13675788. doi: 10.1016/j.arcontrol.2007.04.001.

Thomas Dunker, Günter Radons, and Engelbert Westkämper. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *European Journal of Operational Research*, 165(1):55–69, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2003.01.002.

J-g Kim and Y-d Kim. A branch and bound algorithm for locating input and output points of departments on the block layout. pages 517–525, 1999.

Panagiotis Kouvelis and Michael W Kim. Unidirectional Loop Network Layout Problem in Automated Manufacturing Systems. (May 2020), 1992.

Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976. ISSN 0004-5411. doi: 10.1145/321958.321975. URL https://doi.org/10.1145/321958.321975.

Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. *ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10*, 29(10):1, 2010. ISSN 07300301. doi: 10.1145/1866158.1866203. URL http://portal.acm.org/citation.cfm?doid=1866158.1866203.

Christopher E. Nugent, Thomas E. Vollmann, and John Ruml. An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. *Operations Research*, 16(1):150–173, 1968. ISSN 0030-364X. doi: 10.1287/opre.16.1.150.

Laurent Perron and Vincent Furnon. Or-tools. URL https://developers.google.com/optimization/.

Daniel Scholz, Anita Petrick, and Wolfgang Domschke. STaTS: A Slicing Tree and Tabu Search based heuristic for the unequal area facility layout problem. *European Journal of Operational Research*, 197(1):166–178, 2009. ISSN 03772217. doi: 10.1016/j.ejor.2008.06.028. URL http://dx.doi.org/10.1016/j.ejor.2008.06.028.

Amir Mohammad Esmaieeli Sikaroudi and Kamran Shahanaghi. Facility layout by collision detection and force exertion heuristics. *Journal of Manufacturing Systems*, 41:21–30, 2016. ISSN 02786125. doi: 10.1016/j.jmsy.2016.07.001. URL http://dx.doi.org/10.1016/j.jmsy.2016.07.001.

Kar Yan Tam and Shih Gong Li. A hierarchical approach to the facility layout problem. *International Journal of Production Research*, 29(1):165–184, 1991. ISSN 1366588X. doi: 10.1080/00207549108930055.

James A Tompkins, John A White, Yavuz A Bozer, and Jose Mario Azaña Tanchoco. *Facilities planning*. John Wiley & Sons, 2010.

Ming Jaan Wang, Michael H. Hu, and Meei Yuh Ku. A solution to the unequal area facilities layout problem by genetic algorithm. *Computers in Industry*, 56(2):207–220, feb 2005. ISSN 01663615. doi: 10.1016/j.compind.2004.06.003.