Heuristic Optimization Techniques
Exercise 1

Leon Schwarzäugl

31. October 2023

1 Task 1

Consider a problem in which solutions are represented by binary vectors $x = (x_1, ..., x_n) \in \{0, 1\}^n$ of length n and the 1-flip neighborhood structure is used in a local search with the first improvement step function.
Although such a local search will in practice typically terminate at a local optimum after a reasonably small number of iterations, it may in the worst case require an exponential number.
Construct a function $f(x)$ for which this would be the case when starting the local search from the initial solution $x = (0, \ldots, 0)$. Assume each neighborhood is searched by considering the flips of the variables $x_1, \ldots, x_n$ in this order.
Hint: Start with a fixed small n and try to assign concrete values to all possible solutions in order to maximize the number of needed iterations. Then try to generalize your construction scheme.

The following function was found:

$$f(x) = \sum_{i=1}^{n} x_i(2^i - 1) \prod_{j=i+1}^{n} (-1)[2x_j - 1]$$

This leads to the execution shown in Table 1 (for $n = 3$). Essentially, this function aims to make every flip in an $x_i, i < j, j \in [2, n]$ counteractable by a flip in a $x_j, j > i$, whereas we make use of searching in the order $x_1, \ldots, x_n$ by making sure the gain of flipping a leftwards value is smaller than flipping a value to the right of it while still resulting in an increase. This pattern can also be used for dimensions $n > 3$ by copying the whole scheme downwards while setting the first half of the new column to 0 and the latter half to 1 for each dimension added; the function will lead to 2^n iterations in the worst case.

Table 1: Exponential execution for first improvement step function. The midrules mark the "folding points" for using this pattern in higher dimensions.

x_1	x_2	x_2	$f(x)$
0	0	0	0
1	0	0	1
1	1	0	2
0	1	0	3
0	1	1	4
1	1	1	5
1	0	1	6
0	0	1	7

2 Task 2

Given is the symmetric TSP on the node set $V = \{1, 2, \ldots, n\}$. Assume node 1 is the start and end point of a tour. Let $s = (\pi_1 = 1, \pi_2, \ldots, \pi_k)$ be an arbitrary partial solution in permutation notation containing $k < n$ pairwise different nodes $\pi_1 = 1, \pi_2 \in V, \ldots, \pi_k \in V$ that are already visited in this order.
We apply beam search with $f(\pi) = g(\pi) + h(\pi)$ (see slides), where

1. $g(\pi) = d_{1\pi_2} + d_{\pi_2\pi_3} + \ldots + d_{\pi_{k-1}\pi_k}$ and
2. $h(\pi) = 0$

The value $f(s)$ of a partial solution s is thus the path length from the first node to the last so far visited node of $\pi(g(s))$ and we do not count in any estimate for the further cost-to-go ($h(s) = 0$).
In general, we may expect that a larger beam width will lead to a better or at least equally good solution in comparison to a smaller beam width. This, however, is not always the case. Show this under the above conditions by providing an appropriate concrete TSP instance, select two suitable values for β, and solve the instance with your selected beam widths.

This can be achieved by "luring" the solution away from the optimum by providing $\beta - 1$ nodes that are initially worse, but provide β steps that are better in a subsequent step. This is illustrated in Figure 1 with paths for $\beta = 1$ (cost: $g(s) + 1 + 4 + 5 + 1 + 1 + 7 = g(s) + 19$) and $\beta = 2$ (best cost: $g(s) + 2 + 1 + 2 + 5 + 4 + 30 = g(s) + 44$) in Figure 2.

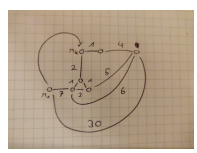

Figure 1: Schematic of a smaller beam size resulting in a better solution. Assume high costs for paths not explicitly described.

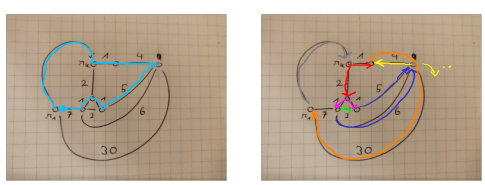

Figure 2: Left: $\beta = 1$. Right: $\beta = 2$ (the path with cost 30 should start at the node where the yellow arrow ends.)

3 Task 3

In the bin packing problem, a set of items $I = \{1, 2, \ldots, n\}$ is given. Each item $i \in I$ has a weight $z_i > 0$. Moreover, bins with a fixed capacity $C > 0$ are given. The goal is to pack all the items in as few bins as possible such that the capacity of each bin is not exceeded.
How can a solution be represented? What local search neighborhood structures do you think make sense? Describe them in terms of the respective move operators that define them.
Design a representation such that your neighborhoods can be efficiently searched. In particular: How can delta evaluation be realized when applying the corresponding move operators to a solution?
Illustrate your representation, neighborhood structures, and ideas for delta evaluation by means of a small example.

We can represent the solution using a vector of length n where each entry corresponds to the respective item and the container that it currently is stored in (for example, for $n = 3, C = 3, z_1 = z_2 = z_3 = 1$ this vector could look like $x = (0, 0, 0)$ with container indices starting at 0). As for useful neighborhood structures, 1-move and 2-exchange come to mind. Depending on the specifics of the problem, operations such as "empty one bin into another" and "empty one bin by distributing its items" could be useful as well. The former two might be implemented by updating the index in the solution vector to the corresponding indices as well as subtracting z_i from the origin bin weight and adding it to the destination bin (for 2-move, these operations are also performed for the swapped item).

4 Task 4

Consider the Quadratic Unconstrained Binary Optimization problem (QUBO) asking for a given real symmetric matrix $A \in \mathbb{R}^{n \times n}$ and a vector of binary decision variables $x = (x_i)_{i=1}^n \in \{0,1\}^n$, to find

$$\max_{x \in \{0,1\}^n} x^T A x.$$

(a) Design your own greedy construction heuristic which tries to come up with a solution of reasonably good quality. In a second step, provide a randomized version of your construction.

(b) As preparation for a local search procedure: What is your solution representation (data structures etc.)? How do you efficiently evaluate the objective function and how do you update it when your current solution is subject to a 1-flip move?

(c) Consider the following problem $\mathcal{P}$: Given a set of numbers $A = \{a_1, \ldots, a_n\} \subseteq \mathbb{N}$, find a "minimally imbalanced" bipartition, i.e., find

$$\min_{\substack{E,F \subseteq A \\ E \cap F = \emptyset \\ E \cup F = A}} \left| \sum_{e \in E} e - \sum_{f \in F} f \right|.$$

Show that each instance of P can be reduced to a suitable QUBO instance whose optimal solution can be translated to an optimal solution of $\mathcal{P}$. Furthermore, design a General Variable Neighborhood Search (GVNS) procedure with the following specifications: Provide three neighborhood structures for the shaking moves and three neighborhood structures for the Variable Neighborhood Descent (VND), where the latter ones rely on the found QUBO representation of the problem P. For each collection of neighborhood structures state which ones contain nested/disjoint neighborhood structures.

4.1 (a)

As the matrix is symmetric, it has $n^2 - \sum_{i=1}^{n-1}$ unique values. Each entry in the solution vector has a bearing on the matrix entries in its own row and column - for example, for $n = 3$ with the matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

we receive

$$a_{11}x_1^2 + 2a_{12}x_1x_2 + 2a_{13}x_1x_3 + a_{22}x_2^2 + 2a_{23}x_2x_3 + a_{33}x_3^2$$

for the total influence of the solution vector. If we want to implement a greedy heuristic, one way is to, in a first step, look for the matrix entry with the biggest absolute value,

$$\max_{i,j} |a_{ij}|.$$

In a second step, we now check if this entry is positive or negative:

1. In the former case, we set one of its influential entries in the solution vector to 1 - for a diagonal element, there is only one choice here, whereas for an off-diagonal element, we decide the element to be set by summing over their respective row- and column values and setting the element in the solution vector that has less negative impact.

2. In the case of the element being negative, we perform the same steps while however setting the corresponding element in the solution vector to 0.

A randomized version of this procedure can be achieved by randomly choosing the element for which to set the corresponding element in the solution vector for from among the m entries in the matrix with the biggest absolute values, for example.

4.2 (b)

The solution is represented as a n-tuple of binary values, $x = (x_i)_{i=1}^n \in \{0,1\}^n$. The initial value of the objective function can be computed by

$$\sum_{i=1}^{n} a_{ii} x_i^2 + \sum_{i=1}^{n} \sum_{j \neq i}^{n} 2a_{ij} x_i x_j,$$

which can be made a marginally more efficient by storing the influences of the off-diagonal elements, $2a_{ij}, i \neq j$, as constant values, which saves evaluations later. Updates can be computed by adding/subtracting the elements in the former expression corresponding to the flipped element in the solution vector. For example, flipping x_1 from 0 to 1 would result in adding $a_{11} + \sum_{j=2}^{n} 2a_{1j} x_j$ to the solution; flipping x_1 from 1 to 0 would result in the subtraction of the same expression.

4.3 (c)

The reduction should result in a matrix that encourages putting small and big elements that are far apart in value, or average elements that are close in valun in the same set, while penalizing putting small and big elements that are close in value in the same set. The behavior for the element spacing can be modeled by taking the product of these elements. Also, the values need to be normalized in some way, or being put in relation to each other. Optimally, the solution sets each contain half of the sum of A.

This can be summarized as (with the matrix elements a_{ij} and the numbers a_i from the set of numbers to be partitioned A)

$$a_{ij} = \begin{cases} a_i(a_i - \sum_{a \in A} a) & \text{if } i = j \\ -(a_i a_j) & \text{if } i \neq j \end{cases}.$$

which's optimal solution will correspond to the optimal solution of $\mathcal{P}$.
As for neighborhood structures for shaking we can use random solution vectors with a 1-, 2-, and 3-flip distance. They are nested here, but we also could take multiple disjoint locations. For the VND, we can use best improvement with 1-, 2-, and 3-flip. These are nested here too.