Appendix: Polynomial reductions

In this appendix we present the polynomial reductions used in our experiments.

Polynomial reduction from VC to SP

We polynomially reduce VC to SP as follows. Given a VC instance, that is, a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of vertexes and $E = \{e_1, \ldots, e_m\}$ is the set of edges, and a natural K, we construct an SP instance (that is, a set of sets and a natural P) from it as follows. We define n sets $S_1, \ldots, S_n$, where each set S_i *represents* the vertex v_i of G in this way: S_i contains an element E_j for each edge e_j which is connected to v_i in G. Besides, let $P = n - K$. We can see that the answer of VC for G and K is *yes* iff the answer for SP for $S_1, \ldots, S_n$ and P is *yes*. Let us prove the implication from left to right. Let A be a set with less than or equal to $K' \leq K$ vertexes of G such that all edges of G are connected to at least one vertex in A. Let $B = V \backslash A$ be the set of $P' = n - K'$ remaining vertexes of G (note that $P' \geq P$). Let $B = \{v_{a_1}, \ldots, v_{a_{n-K'}}\}$. No pair of vertexes of B is connected in G (otherwise, the edge connecting both vertexes would *not* be connected to some vertex in A, which is required). By the construction of sets $S_1, \ldots, S_n$, two sets S_i and S_j are disjoint iff v_i and v_j are not connected in G. Thus, the sets $S_{a_1}, \ldots, S_{a_{n-K}}$ are disjoint to each other. That is, there exist $P' \geq P = n - K$ sets which are disjoint to each other. Proving the implication from right to left is trivial if we reverse the previous steps. Also, note that this reduction takes polynomial time with respect to the size of the instance of VC.

Polynomial reduction from MAX-3SAT to MC

We polynomially reduce MAX-3SAT to MC as follows. Let $\varphi \equiv c_1 \wedge \ldots \wedge c_m$ be a propositional logic formula where each c_i is a disjunctive clause of the form $c_i = l_i^1 \vee \ldots \vee l_i^{k_i}$, where each l_i^j is a literal of the form p or $\neg p$, being p is propositional symbol. We create a polynomial-size instance of MC (i.e. a set of sets and natural numbers K and K') from φ such that the answer for that instance in SC is yes iff there exists a valuation that satisfies at least C clauses of φ. Let $p_1, \ldots, p_n$ be the propositional symbols appearing in φ. For each propositional symbol p_i, we create two sets: S_{p_i}, which contains an element called C_i for each clause c_i in φ which is true if p_i is true (because one of the literals in the clause is p_i), and $S_{\neg p_i}$, which contains an element called C_i for each clause c_i in φ which is true if p_i is false (because one of their literals is $\neg p_i$). Intuitively, taking S_{p_i} represents setting p_i to true, whereas taking $S_{\neg p_i}$ represents setting p_i to false. In order to avoid that we take *both* sets, which would not represent a possible valuation, we include $m + 1$ elements called $P_i^1, \ldots, P_i^{m+1}$ in both S_{p_i} and $S_{\neg p_i}$. Besides, we set $K = n$ and $K' = n \cdot (m+1) + C$. Since there exist $m + 1$ elements of the form C_i, the only way to gather at least $n \cdot (m+1)$ elements is taking all elements of the form $P_i^1, \ldots, P_i^{m+1}$ for all $1 \leq i \leq n$. The only way to pick all of these $n \cdot (m+1)$ elements is taking, for each pair S_{p_i} and $S_{\neg p_i}$, at least one of them. Since only $K = n$ sets can be taken, for each pair at most one of them can be taken. Thus, gathering at least $n \cdot (m+1)$ elements implies forming a valid valuation. Moreover gathering $K' = n \cdot (m + 1) + C$ elements implies that, in addition to all elements of the form $P_i^1, \ldots, P_i^{m+1}$, the sets selection includes C elements of the form C_i, that is, it represents a valuation which makes C clauses of φ true. Since this transformation takes polynomial time, it is a polynomial transformation from MC to MAX-3SAT.

Polynomial reduction from TSP to KN

We polynomially reduce TSP to KN as follows. Let us consider a TSP instance (G, K) where $G = (V, E, c)$ is an undirected weighted graph (where $V = (v_1, \ldots, v_n)$ is the set of vertexes, $E = (e_1, \ldots, e_m)$ is the set of edges where $e_i \in V \times V$ for all $1 \leq i \leq m$, and $c : E \to \mathbb{N}$ is a function associating a cost to each edge), and $K \in \mathbb{N}$. We derive a KN instance from it as follows. We construct $2 \cdot m \cdot n$ pairs of naturals (items) $item_{i,j,k} = (g_{i,j,k}, w_{i,j,k})$ where $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq k \leq 2$. The analogy between TSP and KN is the following: If $item_{i,j,k} = (g_{i,j,k}, w_{i,j,k})$ is included in the KN solution then it will represent, in terms of TSP, that edge $e_j = (v_a, v_b)$ is the i-th edge traversed in the cycle. If $k = 1$ then v_a is the i-th vertex of the cycle and v_b is the $((i \bmod n) + 1)$-th vertex, else (i.e. if $k = 2$) then v_b is the i-th vertex and v_a is the $((i \bmod n) + 1)$-th vertex. The *value* of this item, that is $g_{i,j,k}$, and the *weight* of this item, that is, $w_{i,j,k}$, will be denoted by two binary numbers which are the concatenation of the following segments of consecutive bits:

(A_1) Both the highest bits of $g_{i,j,k}$ and the highest bits of $w_{i,j,k}$ consist of n consecutive segments of $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits each. The definition of these segments is the same for $g_{i,j,k}$ and $w_{i,j,k}$. In both cases, all of these segments are equal to 0 (i.e. all of their bits are 0) but one: The a-th segment is equal to 1 (that is, all bits are 0 but the last one, the least weighted one, which is 1). As we will see later, in any KN solution, for all $1 \leq h \leq n$ the addition of all h-th segments of *values* of all items included in the solution will be required to be *at least* 1, and the addition of all h-th segments of *weights* of all items included in the solution will be required to be *at most* 1. Thus, for each graph vertex, the number of edges *departing* from each vertex in the solution (in the order in which they are traversed in the constructed cycle) will be exactly 1, as it necessarily happens in any TSP solution. Let us note that item values and weights are treated as single (binary) numbers by KN, not as *tuples* of numbers where each one is stored in an independent bit segment of the overall number. However, if we assign appropriate number of bits of the value/weight of the item to each segment, then we will guarantee that *overflows* cannot occur inside each segment, and thus a high value in a segment will never ruin its adjacent segment by producing a carry bit that invades it. Since each item adds up to 1 to each of these segments, no segment can be higher than $2 \cdot n \cdot m$ (which is the total number of items). Thus, overflows are avoided by assigning $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits to each segment. This number of bits is polynomial with respect to the size of the TSP instance. Since there are n segments, the amount of bits needed to represent these segments in values and weights of items is polynomial too.

(A_2) The next highest bits of $g_{i,j,k}$ and the next highest bits of $w_{i,j,k}$, from highest significance to lowest significance, consist of n consecutive segments of $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits each. The definition of these segments is

the same as in A_1, though now the b-th segment is considered instead of the a-th one. That is, edges *arriving* to each vertex are counted, instead of edges departing from each vertex.

(B_1) The next highest bits of $g_{i,j,k}$ and the next highest bits of $w_{i,j,k}$ consist, again, of n consecutive segments of $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits each. The definition of these segments is again the same for $g_{i,j,k}$ and $w_{i,j,k}$. In both cases, all of these segments are equal to 0 (i.e. all of their bits are 0) but one: The i-th segment is equal to 1 (that is, all bits are 0 but the last, which is 1). The addition of i-th segments of all values and weights will count the number of edges which depart from the i-th vertex which is traversed in the formed cycle. So, the difference with A_1 is that, in A_1, we count edges leaving a *given vertex*, though now we count edges leaving the vertex reached at a *given step*, whatever this vertex is.

(B_2) The next highest bits of $g_{i,j,k}$ and the highest bits of $w_{i,j,k}$ consist of n consecutive segments of $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits each. The definition of these segments is the same as in B_1, though now the $((i \bmod n) + 1)$-th segment is considered instead of the i-th one. That is, vertexes reached in the *next step* are counted, instead of vertex left in the current step.

(C) The next bits of both $g_{i,j,k}$ and $w_{i,j,k}$ from highest significance to lowest significance consist of n consecutive segments of $\lceil log_2(2 \cdot (\sum_{q=1}^{m} q) + 1) \rceil$ bits each. Let these segments be called C-segments. In both $g_{i,j,k}$ and $w_{i,j,k}$, all of these segments are equal to 0 (i.e. all of their bits are 0) but two: For both $g_{i,j,k}$ and $w_{i,j,k}$, the i-th segment and the $((i \bmod n) + 1)$-th segment contain the binary representation of numbers a and $n + 1 - b$ respectively (if $k = 1$), or b and $n + 1 - a$ respectively (otherwise, i.e. if $k = 2$). Though other segments of values and weights will guarantee that any KN solution includes, for each graph vertex, *exactly* one item representing an edge reaching it and one item representing an edge leaving it (see A_1 and A_2, respectively), as well as one edge reaching/leaving the vertex located at each cycle step (see B_1 and B_2, respectively), this does not guarantee that all KN solutions will be hamiltonian cycles. For instance, in a graph with vertexes A,B,C,D,E,F, we could take some edges which form the cycles A-B-C-A and D-E-F-D, which does not constitute a hamiltonian graph, though all vertexes are reached/left by exactly one edge and all steps are reached/left by exactly one edge (note that, in the fourth step, we reach A but we leave D). As we will see later, in any KN solution the addition of all C-segments of values of all items included in the solution will be required to be *at least* $n + 1$, and the addition of all C-segments of *weights* of all items included in the solution will be required to be *at most* $n + 1$. By constraints imposed by A_1, A_2, B_1, and B_2, the addition of h-th C-segments of item values and the addition of h-th C-segments of item weights will add exactly $n + 1$ only if the item (edge) selected for h-th step *leaves*, according to the cycle order, some vertex a (so it adds $n + 1 - a$ to h-th C-segment of the total value/weight) and the item (edge) selected for the $((h \bmod n) + 1)$-th step *reaches*, according to the cycle order, the *same* vertex a (so it adds a to the h-th C-segment in the total value/weight). In this case, each edge departs from the vertex reached by the *previous* edge, and so any KN solution must represent a single cycle covering all vertexes of the graph. For each h-th C-segment, even for all $1 \leq j \leq m$ the items $(g_{h,j,1}, w_{h,j,1})$ and $(g_{((h \bmod n)+1),j,2}, w_{((h \bmod n)+1),j,2})$ are included in the KN solution, the C-segment will never add more than $2 \cdot \sum_{q=1}^{m} q$, so $\lceil log_2(2 \cdot (\sum_{q=1}^{m} q) + 1) \rceil$ bits are enough to avoid the overflow between adjacent segments. Since $\sum_{q=1}^{m} q \leq m^2$, we have that $\lceil log_2(2 \cdot (\sum_{q=1}^{m} q) + 1) \rceil \leq \lceil log_2(2 \cdot m^2) \rceil \leq \lceil log_2(m^2) \rceil + 1 \leq 2 \cdot \lceil log_2(m) \rceil + 1$, which is polynomial with respect to the size of the TSP instance. Since there are n C-segments, the amount of bits needed to represent this kind of segments in values and weights of items is polynomial too.

(D) There are no more segments in $w_{i,j,k}$, whereas the next $\lceil log_2(2nmM - 2n \cdot \sum_{q=1}^{m} c(e_q)) + 1 \rceil$ bits of $g_{i,j,k}$ represent $M - c(e_j)$ in binary code, where M is the cost of the edge of G with highest cost. Even if all available items are taken in the KN solution, the addition of these segments in values of all items will not be higher than $2nmM - 2n \cdot \sum_{q=1}^{m} c(e_q)$, so $\lceil log_2(2nmM - 2n \cdot \sum_{q=1}^{m} c(e_q)) + 1 \rceil$ bits are enough to avoid an overflow of this segment into the subsequent segment. Besides, let us note that $\lceil log_2(2nm \cdot M - 2n \sum_{q=1}^{m} c(e_q)) + 1 \rceil \leq \lceil log_2(2nmM) + 1 \rceil$, which is a polynomial number of bits with respect to the size of the TSP instance.

So far we have the definition of all items (edges) of the KN instance derived from the TSP instance. Now, we have to define the minimum required value, L, and the maximum weight capacity of the knapsack, W. They are the binary numbers resulting from the concatenation of the following binary segments of consecutive bits:

($A_{1,2}$) Both the highest bits of L and the highest bits of W consist of $4n$ consecutive segments of $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits each, where all of them are 1 (that is, all bits are 0 but the last one, which is 1).

($B_{1,2}$) Both the highest bits of L and the highest bits of W consist of $4n$ consecutive segments of $\lceil log_2(2 \cdot n \cdot m) + 1 \rceil$ bits each, where all of them are 1 (that is, all bits are 0 but the last one, which is 1).

(C) The next highest bits of L and the highest bits of W consist of n consecutive segments of $\lceil log_2(2 \cdot (\sum_{q=1}^{m} q) + 1) \rceil$ bits each, where all of them are the binary representation of number $n + 1$.

(D) There are no more segments in W, whereas the next $\lceil log_2(2nmM - 2n \cdot \sum_{k=1}^{m} c(e_k)) + 1 \rceil$ bits of L represent $n \cdot M - K$ in binary code, where M is the cost of the edge of G with highest cost.

Let us see that the answer of TSP for G and K is yes iff the answer of KN for pairs $(g_{i,j,k}, w_{i,j,k})$, W, and L is yes.

$\Longrightarrow$: If there is in G a hamiltonian cycle whose sequence of vertexes is $v_{r_1}, \ldots, v_{r_n}$, whose sequence of edges is $e_{j_1}, \ldots, e_{j_n}$, and whose cost $\sum_{i=1}^{n} c(e_{j_i})$ is at most K, then the selection of items $(g_{1,j_1,k_1}, w_{1,j_1,k_1}), \ldots, (g_{n,j_n,k_n}, w_{n,j_n,k_n})$ (where each k_i is 1 if $e_j = (v_{j_r}, v_{j_{r+1}})$ and is 2 if $e_j = (v_{j_{r+1}}, v_{j_r})$) fulfills the conditions that their added weight $\sum_{i=1}^{n} w_{1,j_i,k_i}$ is at most W and their added value $\sum_{i=1}^{n} g_{1,j_i,k_i}$ is at least L. In particular, in the addition of weights and the addition

of costs of all selected items, all bit segments referred in A_1, A_2, B_1, B_2 before must be equal to 1 (because, in the TSP solution, each vertex is reached/left once, and each step is reached/left once too), all segments referred in C before must be $n+1$ (because, in the TSP solution, the vertex reached at each step is the vertex left at the next step, which implies that they add exactly $n+1$), and all segments referred in D add, in the addition of values, $n \cdot M - \sum_{i=1}^{n} c(e_{j_i})$. Since $\sum_{i=1}^{n} c(e_{j_i}) \leq K$, by the construction of L from K we have $n \cdot M - \sum_{i=1}^{n} c(e_{j_i}) \geq L$. Thus, the answer for the KN instance derived from the TSP instance is yes.

$\Longleftarrow$: Let us suppose that, for the KN instance constructed from the TSP instance, the selection of items $(g_{1,j_1,k_1}, w_{1,j_1,k_1}), \ldots, (g_{n,j_n,k_n}, w_{n,j_n,k_n})$ (where each k_i is 1 if $e_j = (v_{j_r}, v_{j_{r+1}})$ and is 2 if $e_j = (v_{j_{r+1}}, v_{j_r})$) fulfills the conditions that their added weight $\sum_{i=1}^{n} w_{1,j_i,k_i}$ is at most W and their added value $\sum_{i=1}^{n} g_{1,j_i,k_i}$ is at least G. This means that, in the addition of weights and the addition of costs of all of these items, all segments referred by A_1, A_2 equal 1, which means that, for all vertexes of G, there is one edge reaching it and one edge leaving it. Besides, in these additions all segments referred by B_1, B_2 equal 1, which means that, for all steps from the first to the n-th, there is one edge reaching this step and one edge leaving it. In addition, in these additions all segments referred by C equal $n+1$, which means that, for all step, the addition of reached vertexes and left vertexes is $n+1$. Since a single edge reaches each step and a single edge leaves each step (by B_1, B_2), we have that, for each step, the vertex reached in the previous step and the vertex left in that step must coincide; otherwise adding $n+1$ would not possible. Besides, since each vertex is reached once and left once (by A_1, A_2), vertexes reached at each step are different to each other, and all vertexes are reached in one step. Thus, we can (uniquely) assign a single vertex to each step. Since edges connect each step with the next, edges represented by selected items necessarily represent form a hamiltonian cycle. Finally, the addition of all segments referred by D in item values is higher than or equal to $n \cdot M - K$, which implies that the cost of the hamiltonian cycle is less than or equal to K. Thus, the answer for the TSP instance from which we derived the KN instance is yes.

Since the size of the KN instance is polynomial with respect to the size of the TSP instance (note that the number of bits required to represent weights and values in each item is polynomial, and the number of items is polynomial too), we conclude that the previous transformation is a polynomial reduction from TSP to KN.

Finally, we illustrate that all segments A_1, A_2, B_1, B_2, C in weights and values are necessary to assure that hamiltonian cycles are formed. Hence, if some segments are removed, then the polynomial reduction would not work, so the proposed reduction is not unnecessarily complex. Let us consider a TSP instance where we have vertexes v_1, v_2, v_3, v_4 and edges $e_1 = (v_1, v_2)$, $e_2 = (v_2, v_3)$, $e_3 = (v_3, v_4)$, and $e_4 = (v_4, v_1)$. In the KN instance derived from this graph, we have 32 items, one for each combination of step, edge, and direction (1 or 2). It is easy to check the following facts:

- $item_{1,1,1}, item_{2,1,2}, item_{3,4,2}, item_{4,4,1}$ fulfill requirements imposed by segments B_1, B_2, C, but they do not form a hamiltonian cycle.
- $item_{1,4,2}, item_{2,4,1}, item_{3,2,1}, item_{4,2,2}$ fulfill requirements imposed by segments A_1, A_2, B_1, B_2, but they do not form a hamiltonian cycle.
- $item_{4,4,2}, item_{4,4,1}, item_{2,2,1}, item_{2,2,2}$ fulfill requirements imposed by segments A_1, A_2, C, but they do not form a hamiltonian cycle.