How to transform a List Coloring problem into a Traveling Salesman one

DRAFT 2001-02-19

Maurice.Clerc@WriteMe.com

Abstract

This paper describes a method to transform a List Coloring Problem (LCP) into a Traveling Salesman Problem (TSP) in polynomial time. Both problems are NP-hard, so it is not surprising such a derivation is possible, but, nevertheless, an explicit method is useful, for any improvement found to solve the TSP can then immediately be used for the LCP. Or, from a negative point of view, if you know a given method is not very good for TSP, you do not even have to try it for LCP: it won't be good, either.

Introduction

The very first idea of the transformation presented here is coming from the study of Frequency Assignment problems. List Coloring of a graph is a classical model for this kind of problem, and a lot of quite powerful algorithms have been designed ([1, 2]). But the Traveling Salesman problem has been (and still is) even more studied, and very good algorithms for large instances habe been designed, particularly for the asymmetric case([3-7]). So it seemed interesting trying to use some of the best TSP algorithms for solving List Coloring problems. In order to do that, it is not enough to have complexity evaluations ([8, 9]): we need an explicit, and, if possible, simple transformation method. We present here a possible one.

From LCP to TSP

Here, "colors" are in fact integer values from a given interval [0,C].

Given a graph G=(V,E), symmetrical, without any loops and multiple edges, and a compatibility matrix M, any mapping $V \xrightarrow{f} \mathbf{N}$ assigning compatible colors to adjacent vertices is a *admissible (vertex) coloring*. Two colors c_{v_i} and c_{v_j}, assigned to the adjacent vertices v_i and v_j, are compatible if they are sufficiently different: $\left|c_{v_i} - c_{v_j}\right| \geq M\left(v_i, v_j\right)$.

Now, if there is a list of colors $L(v)$ associated with every vertex v of G, we obtain a *admissible (vertex) list coloring* if we find an admissible coloring of G assigning to every vertex a color from its own list.

The Travelling Salesman problem is well known. Given a graph G'=(V',E'), in which each edge e has "weight" w, we are looking for a cycle of minimum total weight, crossing any vertex just once (hamiltonian).

The process of transforming a LCP into a TSP has three phases:

- Find a "guiding" cycle of G using any edge or its symmetrical just once. You perfectly may use several times any node and any edge, so it is easy. The least would be the best, but there is no need to use a sophisticated (and expensive) algorithm which could change the complexity level of the whole problem. This guiding cycle will ensure all (symmetrical) constraints of the compatibility matrix will be taken into account.
- Replace all nodes by an Input/Output/(Reinput-reoutput) cycle (see below). So, we begin to build a graph G' which "contains" all possible colors for all nodes of G, ensuring that at least one minimum hamiltonian cycle in G' will affect just one color to any node of G.

- Generate the weighted edges *E'* for *G*', using *E* of *G*. This takes the compatibility matrix into account. Then, solving the TSP for *G*' will give the best coloring solution for *G* (assuming either we find all the solutions or the algorithm is specifically modified, as seen below).

To explain and illustrate this process, we use here a very simple graph (see Figure 1), the two same possible colors {1,2} for each node, and the only constraint is that two adjacent nodes can not have the same color. So, the compatibility matrix is defined by

$$\begin{cases} (v_i, v_j) \in E \Rightarrow M(v_i, v_j) = 1 \\ (v_i, v_j) \notin E \Rightarrow M(v_i, v_j) = 0 \end{cases}$$

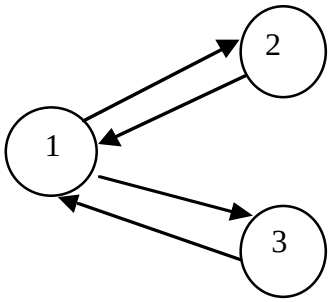

Figure 1. Graph to color with at most two colors.

Phase 1: Finding a guiding cycle

To build a guiding cycle, an algorithm can easily be written([8]), using these simple rules:

a) begin at edge 0,
b) to add a new arc (v_i,v_j), first try to choose one so that (v_j,v_i) has not been used, and, second, go towards the adjacent node the less already used,
c) do not use twice the same edge,
d) continue as long as as you can and as long there is an edge (v_i,v_j) for which neither (v_i,v_j) nor (v_j,v_i) are used.

Usually, it is enough. But sometimes, because of c), such an algorithm can not find a whole cycle. To complete it, we may have to use a stupid but sure one for any symmetrical edge not taken into account (that is to say neither (v_i,v_j) nor (v_j,v_i) are used):

- go back to 0 from the last node found,
- go from 0 to v_i,
- add (v_i,v_j),
- go back to 0.

Note that it works only if the graph is not disconnected into more than one component. But this is not a problem, for each component could be thenconsidered independently.

Figure 2 shows a the guiding cycle found for our example.

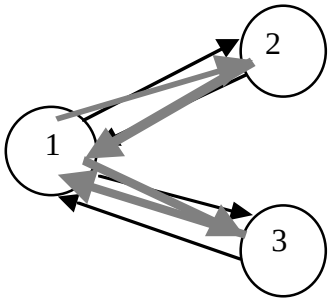

Figure 2. A guiding cycle (start node: 1). Node 2 is used twice.

Phase 2: Inserting I/O/R cycles

Let us consider a vertex *v* of the guiding cycle. It has a color list $L(v) = \{c_1, ..., c_{|L|}\}$. The idea is to replace *v* by a subgraph v' whose nodes *c'* represent these colors. These nodes are divided in at least two sets, and more if *v* is used more than one time in the guiding cycle. Each set is exactly equal to *L*(*v*):

- input nodes $I(v) = \{c'_{I,1}, ..., c'_{I,|L|}\}$
- output nodes $O(v) = \{c'_{O,1}, ..., c'_{O,|L|}\}$
- optional reinput/reoutput nodes (one set for each time the node *v* is reused in the guiding cycle) $R_k(v) = \{c'_{R_k,1}, ..., c'_{R_k,|L|}\}$

Now, edges in *v'* (all weights equal to zero) are defined so that the final subgraph has the following features, relative to a TSP algorithm. If the salesman goes into *v*' using the input node $c'_{I,j}$, then he must be able to find a minimum global hamiltonian cycle by:

e) going just once through any other node of *I*(*v*) (input nodes)
f) going just once through any reinput/reoutput node, except the $c'_{R_k,j}$ ones (same color)
g) going just once through any node of *O*(*v*) (output nodes)
h) leaving *v*' by the node $c'_{O,j}$ (same color as the one of the input point).

Figure 3 shows a possible structure for such an I/O/R-cycle. Note that, as we will see, all outcoming edges have a positive weight.

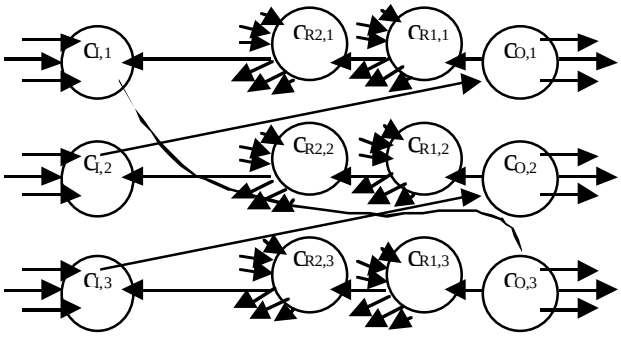

Figure 3. The I/O/R-cycle structure (here with two reinput sets).

Phase 3: Weighting edges

As we have seen, in G', all edges inside I/O/R-cycles are set to zero.

Outside the I/O/R-cycles, each edge is coming from an output (O) or reinput/reoutput (R) node and going to an input (I) or reinput/reoutput (R) node.
Let c_{X,j,v_p} be the X -node (X=I or O or R) for color *j* and for the I/O/R-cycle corresponding to the vertex v_p of *G*. We progressively generate the weighted edges of *G*' according to the guiding cycle.

We generate an edge from c_{X,j,v_p} to c_{Y,j,v_q} if and only if the edge (v_p,v_q) belongs to the guiding cycle found in Phase 1, and using the following rules:

i) If an output node of the subcycle v'_p has not yet been used, use it (X=O).
j) If all output nodes of the subcycle v'_p have already been used, and a reinput/reoutput node has not yet been used as an output, use it (X=R).
k) If an input node of the subcycle v'_q has not yet been used, use it (Y=I).
l) If all input nodes of the subcycle v'_q have already been used, and a reinput/reoutput node has not yet been used as aninput, use it (Y=R).

These rules ensure at least one minimum hamiltonian cycle of *G*' will use one and just one color in each subcycle *v'*.

For the weight *w*, a "binary" version for such an edge in *G*' can be

$$w\left(\left(c_{X,i,v_p},c_{Y,j,v_q}\right)\right)=2 \text{ if } \left|c_{X,i,v_p}-c_{Y,j,v_q}\right|>M(v_p,v_q)$$
$$=1 \text{ else}$$

This weighting just count how many times a constraint is not satisfied. We use values 2 and 1 instead of 1 and 0 , for it may improve some TSP algorithms (which progressively build a feasible path), particularly the ones which use a "lowest first" method, Anyway, as noted below, it is better to specially modify the TSP algorithm you use.

Another weighting, more interesting if there is no perfect solution, for it evaluates how "far" we are for the optimum we want, is

$$w\left(\left(c_{X,i,v_p},c_{Y,j,v_q}\right)\right)=1+\left|c_{X,i,v_p}-c_{Y,j,v_q}\right|-M(v_p,v_q) \text{ if } \left|c_{X,i,v_p}-c_{Y,j,v_q}\right|>M(v_p,v_q)$$
$$=1 \text{ else}$$

Solving the TSP

A solution tour is called *admissible* if, for a given I/O/R-cycle, the input node, the output node and, eventually, the reinput/reoutput nodes have the same color.
Let $d^+_{\min}$ and $d^+_{\max}$ be the minimum and maximum number of adjacent vertices for a given vertex of *G*. We have then, for the graph size (number of nodes of *G*'),
$d^+_{\min}\left(\max|L(v)|\right)|V| \le |V'| \le d^+_{\max}\left(\max|L(v)|\right)|V|$. On the one hand, if d^+_{max} is quite high, and as the computer time needed by any TSP algorithm is rapidly increasing with the graph size, it may seem the transformation LCP into TSP if then not wortwhile. But, on the other hand, the result is a *special* TSP: when looking for a hamiltonian cycle, as soon as you use an input node of a subgraph *v*', you know in advance a possible whole sequence of next nodes before to leave *v*'.
So, in practice, you can easily specifically modify a TSP algorithm in order to find only admissible tours and it is then almost like a TSP of size $|V|$. Note that this is an asymmetric problem, easiest to solve than a symmetric one([5]).

Figure 4 shows the final *G*' graph for our example, and a schematic view of a mimimal hamiltonian cycle (just input and output nodes of I/O/R-cycle are "hightlighted"). There are others solutions which are not admissible, for they use input/output nodes of different colors for the same I/O/R-cycle, for example (1,6,4,11,14,12,13,2,5,8,9,7,10,3,1).

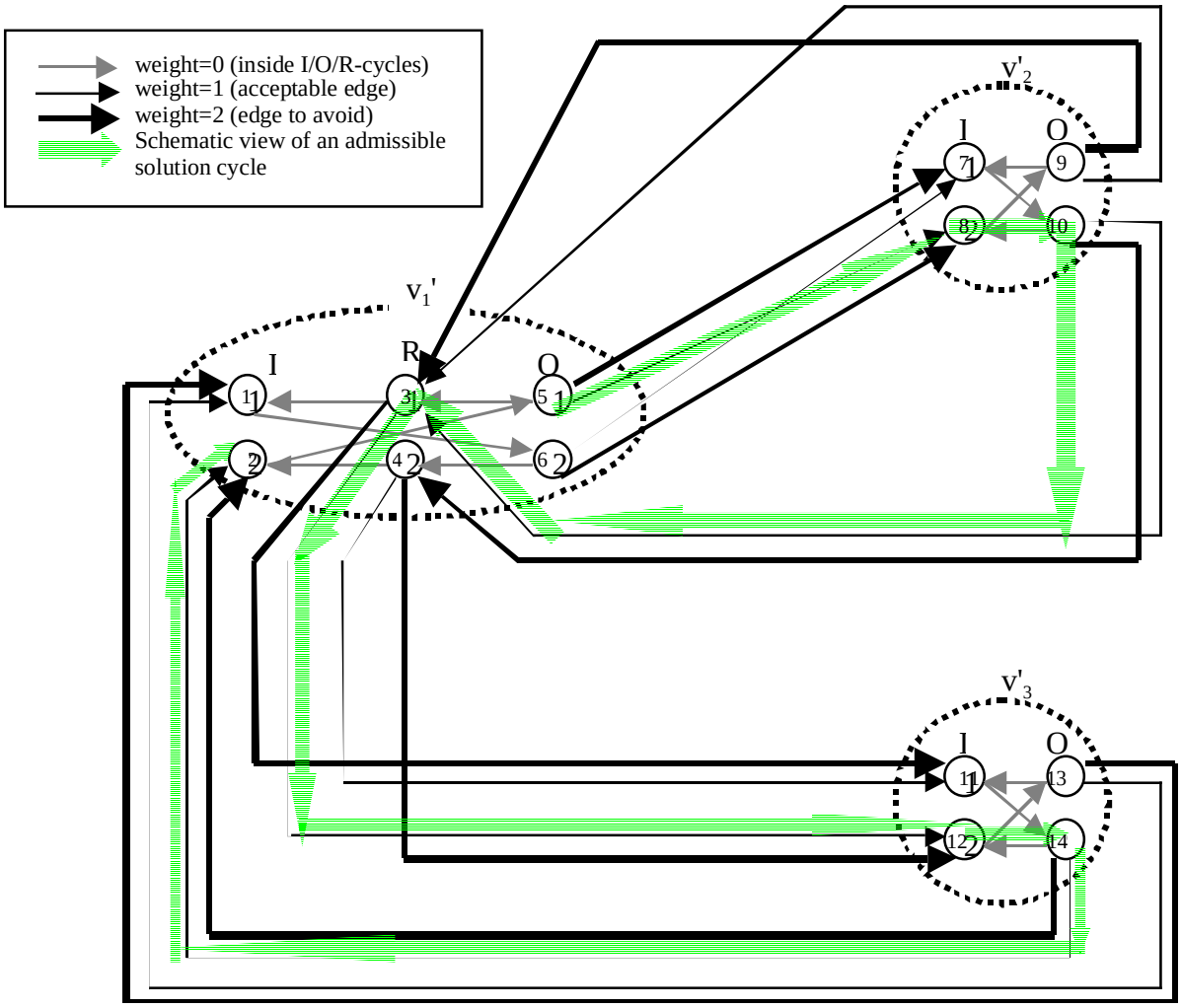

Figure 4. The whole graph G' for the Traveling Salesman Problem.

Back to the LCP

Extracting a solution for the LCP is now straightforward: we follow the solution cycle in *G*', but keeping, for any subgraph *v*', just the input node found on the way. For our example, it gives the admissible coloring (1,2,2).

References

[1] A. Eisenblätter, M. Grötschel, and A. M. C. A. Koster, “Frequency Planning and Ramifications of Coloring,” Konrad-Zuse-Zentrum für Informationstechnik Berlin. ZIP-Report OO-47, 2000.
[2] E. Malesinska, “Graph-Theoretical Models for Frequency Assignment Problems,” : Technischen Universität Berlin., 1997.
[3] Z. He, C. Wei, B. Jin, W. Pei, and L. Yang, “A New Population-based Incremental Learning Method for the Traveling Salesman Problem,” presented at Congress on Evolutionary Computation, Washington D.C., 1999.
[4] K. Helsgaun, “An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic,” Department of Computer Science,Roskilde University, Denmark 1997.
[5] M. Bellmore and J. C. Malone, “Pathology of traveling-salesman subtour-elimination algorithms,” *Oper. Res.*, vol. 19, pp. 278-307, 1972.
[6] G. Carpaneto, M. Dell'Amico, and P. Toth, “Exact solution of large-scale, asymmetric traveling salesman problems,” *ACM Transactions on Mathematical Software*, vol. 21, pp. 394-409, 1995.
[7] D. L. Miller, Pekny, and J. F., “Exact solution of large asymmetric traveling salesman problems,” *Science*, vol. 251, pp. 745-761, 1991.
[8] M. Gondran and M. Minoux, *Graphes et algorithmes*: Eyrolles, 1979.
[9] W. G. Macready and D. H. Wolpert, “What Makes An Optimization Problem Hard?,” The Santa Fe Institute 1996.