DIFFERENTIAL EVOLUTION FOR SMALL TSPs WITH CONSTRAINTS

Štefan Peško[1]

Summary: This paper presents evolution algorithm for solving small (up to 32 nodes) traveling salesman problems with constraints. This new differential evolution algorithm with only two parameters - the size of the population and the size of the generations use the Lehmer code of the permutation for the representation of populations. Experience with small instances of TSP with time windows and deadline TSP are discussed.

Key words: evolution algorithm, traveling salesman problem , time constraints.

1 Introduction

The Travelling Salesman Problem (TSP) is the search for the shortest tour that visits a given set of nodes exactly once. The TSP is one of the oldest optimization problem since TSPs are frequent components of optimization problems.

As noted in [1] the existence of exchange neighbourhood structures of TSP provides very good heuristics, the existence of various relaxations of the TSP provides very precise lower bounds and structure of TSP polytope can be exploited with powerful branch and cut method solving problems as large as 6000 nodes.

Why are then small TSPs interesting? TSPs are often modeled in the real vehicle routing problems (VRP), which could be seen as multiple TSP when one has to find a tour for a vehicle such that all nodes are visited by one tour. However even though the size of a VRP may be very large, the number of visits that a vehicle can make by one trip is bounded by capacity of vehicle or legal constrains to relative small number. In the case of an insertion heuristics using an exact algorithm for the small TSP instead of a local optimization heuristics translated into a global saves of 1%. On the other hand, one running of the VRP for 1000 nodes

[1] RNDr Štefan Peško, CSc., University of Žilina, Faculty of Management Science and Informatics, Department of mathematical method, Univerzitná 8215/1, 01026 Žilina, Slovak Republic, tel.: +421 415 134 275, E-mail: pesko@frcatel.fri.utc.sk1

may translate into 10000 running of the subproblems and thus to solve small TSPs efficiently is important.

The key requirement for resolution of small TSPs is flexibility -- the ability to solve problems with side constraints. One can add time windows to TSP nodes that indicate a interval in which the node must be visited. Other instances include changing travel time between one or many time windows. Interesting case is probabilistic TSP [2] where nodes are visited with given probability.

2 TSP with constraints

The TSP with constraints can be stated in simple terms: Given an real $(n+1) \times (n+1)$ distance matrix $\boldsymbol{D}=(d_{ij})$, find a permutation π of the set $N=\{1,2,\ldots,n\}$ that minimise the function

$$c(\pi) = d_{0,\pi(1)} + \sum_{k=1}^{n-1} d_{\pi(k),\pi(k+1)} + d_{\pi(n),0} + K \cdot m(\pi), \qquad (1)$$

where $m(\pi)$ is measure of infeasibility of permutation π considering the same type of the side constraints and K is positive penalty constant. The size of the TSP with constraints we define equal $n+1$. The travelling salesman tour (TSP tour) is here in the form of the cycle $0\rightarrow\pi(1)\rightarrow\pi(2)\rightarrow\ldots\rightarrow\pi(n)\rightarrow 0$ where $\pi(k)$ is k^{th} visited node in this tour from the start node 0. Note that for the feasible permutation π we define $m(\pi)=0$.

Example 1.: For the instance of the TSP with *time windows* (TSPTW) intervals $[\tau_i^R,\tau_i^D]$ are given for nodes i from N where τ_i^R is release time and τ_i^D is deadlines time. The distance d_{ij} represents sum of the minimum time of the trip from node i to node j and the processing time in node j. The TSPTW tour is feasible (in the simple case) if each arrival t_i is in interval $[\tau_i^R,\tau_i^D]$. Formally the TSPTW tour is feasible if we find a solution t_i of following conditions:

$$\begin{aligned} &\tau_i^R \le t_i \le \tau_i^D \quad \textit{for } i \in N, \\ &t_0 + d_{o,\pi(1)} \le t_{\pi(1)}, \qquad (2) \\ &t_{\pi(i-1)} + d_{\pi(i-1),\pi(i)} \le t_{\pi(i)} \quad \textit{for } i \in N - \{1\}. \end{aligned}$$

Then we can define the measure of infeasibility $m(\pi)$ of the TSPTW tour as a minimum total time out of constraints

$$m(\pi) = \min_{t_1, t_2, \ldots, t_n \geq 0} \sum_{i \in N} (A_i + B_i) \;\; where \;\; \pi(0) = 0 \;\; and$$
$$A_i = \max\{0, t_i - \tau_i^D, \tau_i^R - t_i\}, \;\; B_i = \max\{0, t_{\pi(i-1)} + d_{\pi(i-1),\pi(i)} - t_{\pi(i)}\}. \tag{3}$$

Example 2.: For the instance of the *deadline TSP* (DTSP) only deadlines τ_i^D for each visited node i from N and the start time τ_0 for the start node 0 are given. The distance d_{ij} is interpreted as for the TSPTW. The DTSP tour is feasible if each arrival t_i to node i is possible before their deadline τ_i^D. The DTSP is special case of the TSPTW when $\tau_i^R = \tau_0$.

3 Lehmer code

The set of all permutation of the set N we note S_n. In this paper we offer an algorithm where a permutation $\pi = < \pi(1), \pi(2),\ldots, \pi(n) >$ is represented via its Lehmer code $L(\pi)$.

General works in discrete mathematics and theoretical computer science deal with effective ways to represent permutation . A pioneer of this matter is Lehmer [3] who associates S_n and L_n, where L_n is subset of $\{0,1,\ldots,n\text{-}1\}^n$. There are several ways to establish this one-to-one correspondence. The most classical of them is following: *The Lehmer code of the permutation π is a sequence of the numbers*

$$L(\pi) = \left(l_1(\pi), l_2(\pi), \ldots l_n(\pi)\right), \;\; l_i(\pi) = \left|\{j > i : \pi(j) < \pi(i)\}\right|, \tag{5}$$

where $l_i(\pi)$ is the number of entries to the right of $\pi(i)$, which are smaller. It is not difficult to see how $\boldsymbol{\pi}$ can be reconstructed from the code $L(\pi)$:

$$\pi(k) = N_k[l_k(\pi) + 1)], \;\; where \;\; N_k = N - \{\pi(1), \pi(2), \ldots, \pi(k-1)\},$$

with respect to the natural order of the sets N_k..

Example 3.: From definition (5) we have *L(<4,6,2,5,3,1,8,7>) = (3,4,1,2,0,1,0).* We show *that L^{-1}((3,4,1,2,1,0,1,0).) = (<4,6,2,5,3,1,8,7>).*

$\pi(1)=N_1[\mathbf{3}+1]=4$, $N_1=\{1,2,3,4,5,6,7,8\}$, $\pi(2)=N_2[\mathbf{4}+1]=6$, $N_2=\{1,2,3,5,6,7,8\}$,
$\pi(3)=N_3[\mathbf{1}+1]=2$, $N_3=\{1,2,3,5,7,8\}$, $\pi(4)=N_4[\mathbf{2}+1]=5$, $N_4=\{1,3,5,7,8\}$,

$\pi(5)=N_5[\mathbf{1}+1]=3$, $N_5=\{1,3,7,8\}$, $\pi(6)=N_6[\mathbf{0}+1]=1$, $N_6=\{1,7,8\}$,

$\pi(7)=N_7[\mathbf{1}+1]=8$, $N_7=\{7,8\}$, $\pi(8)=N_8[\mathbf{0}+1]=5$, $N_4=\{7\}$.

The Lehmer code establishes a bijection between S_n and the set of sequences L_n. Now we can define operations $\pm$ with permutations via the Lehmer code. Let π and ψ are in S_n with the Lehmer codes $L(\pi)$ and $L(\psi)$. Then we define - for our algorithm – permutations

$$\pi+\psi = L^{-1}\left([l_1(\pi)+l_1(\psi)]\bmod n,\dots,[l_k(\pi)+l_k(\psi)]\bmod(n-k+1),\dots,0\right),$$
$$\pi-\psi = L^{-1}\left([l_1(\pi)-l_1(\psi)]\bmod n,\dots,[l_k(\pi)-l_k(\psi)]\bmod(n-k+1),\dots,0\right).$$

4 Differential evolution

Differential evolution (DE) [4] is a new heuristical approach for minimising real-valued multimodal objective functions. In DE algorithm is a population based on algorithm like genetic algorithm using the same operators; crossover, mutation and selection. The main difference in constructing better solution is that genetic algorithm rely on crossover while DE relays on mutation operation. In our DE algorithm no crossover is used.

We now describe our version of the DE algorithm for the TSPs with constraints.The goal is to find $\pi^* = argmin\{c(\pi): \pi \text{ in } S_n\}$.The DE works with two populations $\boldsymbol{P}$ and $\boldsymbol{M}$ of size M. For an old population $\boldsymbol{P}=\{ \pi_1, \pi_2,,\dots,\pi_M \}$ a mutant population $\boldsymbol{M}=\{ \mu_1, \mu_2,,\dots,\mu_M \}$ is generated by adding the difference between two elements of $\boldsymbol{P}$ and third element of $\boldsymbol{P}$ according to the rule

$$\mu_t = \arg\min\{c(\pi_x+(\pi_y-\pi_x)),c(\pi_x+(\pi_y-\pi_e)),c(\pi_e)\}, \qquad (6)$$

where x,y,z are integers taken at random from the set $\{1,2,\dots,M\}$, mutually different and different from running index t. The random permutation π_e has Lehmer code $L(\pi_e) = (rand(n),rand(n-1),\dots,rand(2),0)$. We assume a uniform probability distribution function $rand(k)$ for numbers from the set *{0,1,\dots,k-1}*.

The new population is created by replacing some element of the old $\boldsymbol{P}$ to better element of the mutant population. More formally the DE algorithm is writen on the figure 1 as a function DEtsp().

```
function DEtsp(𝔻,size_pop ,size_gen)
(* Initialisation*)
  n = size(𝔻)
  for t = 1 to size_pop do
      L(π_t) = (rand(n), rand(n − 1), . . . , rand(2), 0)
(* Evolution of generations*)
  for noGen = 1 to size_gen do begin
      generate the mutant population M
      for t = 1 to size_pop do
          if c(μ_t) < c(π_t) then
              π_t = μ_t
      end
(* Selection of best solution*)
  π_DE = π_1
  for t = 2 to size_pop do
      if c(π_t) < c(π*) then
          π_DE = π_t
return π_DE
```

Figure 1: DE algorithm

Function DEtsp() has two fixed parameters size of the population (size_pop) and the size of the generations (size_gen) and return the best solution of the last generation π_{DE}. The initial population is chosen randomly. The Lehmer codes are used in calculation of permutations of mutant population defined by relation (6).

5 Experimental results

Presented DE algorithm was implemented in Python 2.3+. The corresponding program was tested on the small instances of DTSP and TSPTW from selected instances for asymmetric TSPTW [5]. The symmetric instances of the problems (with symmetric distance matrix ***D'*** are generated by a simple rule from the distance matrix ***D*** of asymmetric instances via rule $d_{ij}' = min(d_{ij},d_{ji})$.

The results of experiment with the size of the population *size_pop =2. n*, size of the generations *size_gen =1000. n* and penalty constant $K=n.max\{c_{ij}\}$ are presented for the asymmetric and the symmetric instances of the DTSP and and for the TSPTW in Table1.

Table1.: Rune time of the DE algprithm (*optimal solution)

Instances [5]	Size of TSP	DTSP symmetric	DTSP asymmetric	TSPTW symetric	TSPTW asymmetric
rbg10a	11	54s	43s	123s	107s*
rbg017	18	96s	87s	102s	203s
rbg021	22	128s	111s	256s	245s*
rbg27	28	468s	438s	558s	561s*
rbg031a	32	602s	538s	668s	628s*

The first result is that DE solve a small TSP with constraints can be very robust and efficient manner.

This paper is supported by the grant VEGA 1/0490/03

Reference literature

1. CASEAU Y., LABURTHE F.: Solving Small TSPs with Constraint. *In Proc. of the 14 st International Conference on Logic Programming,* The MIT Press, 1997, pp. 1-15.
2. JANÁČEK J., HURTÍK J.: An Inpact of Improvement-exchange Heuristics to Quality of Probabilistic TSP Solution. Komunikécie/Communications 4/2003, Scientific letters of the university of Žilina, 2003, pp. 27-35.
3. LEHMER H.: Teaching combinatorial tricks to a computer, *In Proc. Sympos. Appl. Math. Combinatorial Analysis,* Vol. 10, Amer. Math. Soc., Providence, R.I., 1960, pp. 179-193.
4. STORN R., PRICE K.: Differential evolution - A Simple and Efficient Heuristic Strategy for Global Optimisation over Continuous Spaces, Journal of Global Optimisation, Vol. 11, 1997, pp. 341-375.
5. ASCHEUER N.: Instances of the Asymmetric Traveling Salesman Problem with Time Windows (ATSPTW)}, available on http://elib.zib.de/pub/mp-testdata/tsp/atsptw/text.en.html