Acta Cybernetica **12** (1995) 209-216.

On a tour construction heuristic for the asymmetric TSP

I. Bartalos * T. Dudás† B. Imreh *

Abstract

In this paper we deal with a new tour construction procedure for the asymmetric traveling salesman problem. This heuristic is based on a new patching operation which joins three subtours together. Regarding the efficiency of this procedure, we present an empirical analysis.

It is well-known that the assignment problem is a relaxation of the traveling salesman problem (TSP). Thus, if the optimal assignment is a tour, then it is also an optimal solution of the TSP. Otherwise it consists of disjoint cycles. For some special cases of the TSP, these cycles can be patched into an optimal tour. The first algorithm based on this technique was presented by P. C. Gilmore and R. E. Gomory in [1]. Their idea was involved in several procedures solving different special TSP models. A nice discussion of the well-solved cases can be found in [3].

For the general TSP there is no effective procedure to convert the optimal assignment into an optimal tour. Nevertheless as the computational experiments of E. Balas and P. Toth (see [3]) show, the lower bound resulting from the assignment problem is often very tight. On the other hand, the number of the cycles of the optimal assignment is not large in general. These facts suggest that a suitable subtour patching method may result in a good TSP heuristic. The first such heuristic for the asymmetric TSP was presented by R. M. Karp [5] and a similar one was given in [3]. In both cases the method converts the optimal assignment into a tour by a sequence of patching operations, each of which joins two cycles together. Our procedure is based on such a patching operation which joins three cycles together if the number of the cycles of the optimal assignment is not large. Algorithms joining four or more cycles in each step have too high complexity.

As far as the number of the cycles is concerned, it is known that if we choose a permutation of $\{1,\ldots,n\}$ at random, then the expected number of its cycles is $\log(n)$ (see e.g. [4]). We can, however, restrict our consideration for permutations without a fixpoint. Indeed, without loss of generality we may assume that the optimal assignment does not contain diagonal elements. Such an assignment can be achieved by choosing suitably large coefficients in the diagonal of the cost matrix.

*Department of Informatics, József Attila University, P.O. Box 652, H6701 Szeged, Hungary

†Institute of Mathematics, Technical University of Graz, Kopernikusgasse 24, A-8010 Graz, Austria.

The expected number of cycles in a randomly chosen permutation without a fixpoint has not been calculated as yet. Here we prove that the approximate value of this number is still $\log(n)$.

To start with, let us denote by R_n the set of permutations without a fixpoint on the set $\{1,\dots,n\}$ and let $|R_n| = r_n$. It is known (see [2] p.10) that

$$r_n = n!(1 - \frac{1}{1!} + \dots + (-1)^n \frac{1}{n!}) \approx \frac{n!}{e}.$$

Let $2 \le k \le n-2$ be an arbitrary fixed integer and $i \in \{1,\dots,n\}$ where $n \ge 5$. Let us count those permutations of R_n in which i is contained in a cycle of length k. There are $\binom{n-1}{k-1}$ possible ways to choose the elements of this cycle and $(k-1)!$ ways to order them. The number of the permutations without a fixpoint of the remaining $n-k$ elements is r_{n-k}. Therefore, the number we seek is

$$\binom{n-1}{k-1}(k-1)!r_{n-k}.$$

It is obvious that the number of the permutations in which i is contained in a cycle of length n is $(n-1)!$.

Now let us consider R_n as a sample space in which each permutation is assigned a probability $1/r_n$. For any $i \in \{1,\dots n\}$ and $k \in \{2,3\dots,n-2,n\}$, let us denote by $\xi_i^{(k)}$ the random variable on R_n for which

$$\xi_i^{(k)} = \begin{cases} 1 & \text{if } i \text{ is contained in a } k\text{-cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

Using the numbers determined above, we obtain

$$E(\xi_i^{(k)}) = \binom{n-1}{k-1}(k-1)!\frac{r_{n-k}}{r_n} \quad \text{if} \quad 2 \le k \le n-2 \quad \text{and}$$

$$E(\xi_i^{(n)}) = (n-1)!\frac{1}{r_n}.$$

Now $\xi_1^{(k)} + \dots + \xi_n^{(k)}$ is the number of points which are contained in k-cycles, and $\eta_k = \frac{1}{k}(\xi_1^{(k)} + \dots + \xi_n^{(k)})$ is the number of k-cycles. The expected value of η_k is

$$E(\eta_k) = \frac{n}{k}\binom{n-1}{k-1}(k-1)!\frac{r_{n-k}}{r_n} = \frac{n!}{k(n-k)!}\frac{r_{n-k}}{r_n} \quad \text{if} \quad 2 \le k \le n-2 \quad \text{and}$$

$$E(\eta_n) = \frac{(n-1)!}{r_n}.$$

Then $\eta_2 + \eta_3 \dots + \eta_{n-2} + \eta_n$ is the number of cycles and for the expected value u_n of this number, we obtain

$$u_n = \frac{1}{r_n}\left((n-1)! + \sum_{k=2}^{n-2} \frac{n! r_{n-k}}{k(n-k)!}\right).$$

Now substituting r_{n-k} and exchanging the order of the summation, we get that u_n is equal to

$$\frac{1}{r_n}\left((n-1)! + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-i} \frac{n!}{k}\right) =$$

$$\frac{n!}{r_n}\left(\frac{1}{n} + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k} - \sum_{i=3}^{n-2} \frac{(-1)^i}{i!} \sum_{k=n-i+1}^{n-2} \frac{1}{k}\right).$$

Let

$$w_i = \frac{1}{i!} \sum_{k=n-i+1}^{n-2} \frac{1}{k}, \quad i = 3, \ldots, n-2.$$

Then

$$u_n = \frac{n!}{r_n}\left(\frac{1}{n} + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k} + \sum_{i=3}^{n-2} (-1)^{i+1} w_i\right).$$

It is easy to see that $w_3 > \ldots > w_{n-2} > 0$, and so,

$$0 < \sum_{i=3}^{n-2} (-1)^{i+1} w_i \leq w_3.$$

On the other hand,

$$\frac{1}{n} + \frac{1}{3!}\frac{1}{n-2} < \frac{7}{6(n-2)}.$$

Therefore,

$$\frac{n!}{r_n}\left(\sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k}\right) < u_n < \frac{n!}{r_n}\left(\frac{7}{6(n-2)} + \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \sum_{k=2}^{n-2} \frac{1}{k}\right).$$

If n is large enough, then

$$\frac{n!}{r_n} \approx e, \qquad \sum_{i=2}^{n-2} \frac{(-1)^i}{i!} \approx e^{-1}, \qquad \frac{7e}{6(n-2)} + \sum_{k=2}^{n-2} \frac{1}{k} \leq \log(n-2),$$

and so, $\log(n-1) - 1 \leq u_n \leq \log(n-2)$.

Now we recall the definition of the patching operation (see [3]). For this reason let us consider an asymmetric TSP of n cities with cost matrix $\mathbf{C}$ and let φ denote

an optimal assignment which is not a tour. For the sake of notation simplicity, we shall identify every cycle with the set of its cities. Assuming that φ has no fixpoint, let i and j be two cities that occur in two distinct cycles U and V. Then deleting the arcs $(i,\varphi(i))$, $(j,\varphi(j))$ and inserting the arcs $(i,\varphi(j))$, $(j,\varphi(i))$, we join U and V into a new cycle, and thus obtain a new assignment $\bar{\varphi}$. This operation is called the (i,j)-*patching operation*. For the cost of the new assignment, we get

$$z(\bar{\varphi}) = z(\varphi) + c_{i\varphi(j)} + c_{j\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)}\,.$$

The difference $z(\bar{\varphi}) - z(\varphi) = c_{i\varphi(j)} + c_{j\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)}$ is called the *patching cost of the* (i,j)*-patching operation*. The minimal patching cost with respect to U and V is

$$\Delta(\varphi,U,V) = \min\{c_{r\varphi(s)} + c_{s\varphi(r)} - c_{r\varphi(r)} - c_{s\varphi(s)} : r \in U,\, s \in V\}\,.$$

This means that the cycles U and V can be joined into a new cycle with cost $\Delta(\varphi,U,V)$, but they cannot be joined with a lower cost by any (i,j)-patching operation. Therefore we say that the *2-patching cost of the cycles* U *and* V is $\Delta(\varphi,U,V)$.

Now we are ready to present the algorithm developed in [3].

The 2-patching algorithm

Step 1. Determine an optimal assignment φ.

Step 2. If the current φ is a cyclic permutation then terminate.
Otherwise go to Step 3.

Step 3. Choose two cycles U and V of φ such that $|U|$ and $|V|$ are maximal. Calculate $\Delta(\varphi,U,V)$ and let $i \in U$, $j \in V$ such cities for which the patching cost is $\Delta(\varphi,U,V)$. Perform the (i,j)-patching operation and consider the new assignment as the current φ. Return to Step 2.

Extending the patching idea for three cycles, we can define the (i,j,k)-*patching operation* as follows.

Let i,j,k be three cities which occur in three distinct cycles U,V,W. Then deleting the arcs $(i,\varphi(i))$, $(j,\varphi(j))$, $(k,\varphi(k))$ and inserting the arcs $(i,\varphi(j))$, $(j,\varphi(k))$, $(k,\varphi(i))$, we join U, V and W into a new cycle. The patching cost of this operation is

$$c_{i\varphi(j)} + c_{j\varphi(k)} + c_{k\varphi(i)} - c_{i\varphi(i)} - c_{j\varphi(j)} - c_{k\varphi(k)}\,.$$

The minimal patching cost with respect to U,V,W is

$$\Theta(\varphi,U,V,W) = \min\{c_{r\varphi(s)}+c_{s\varphi(t)}+c_{t\varphi(r)}-c_{r\varphi(r)}-c_{s\varphi(s)}-c_{t\varphi(t)}:$$
$$r \in U, s \in V,\ t \in W\}.$$

Then there are cities $i \in U$, $j \in V$, $k \in W$ such that the cycles U, V, W can be joined into a cycle with cost $\Theta(\varphi, U, V, W)$. This cost is called the *3-patching cost of the cycles* U, V, W.

Now we present a procedure based on the introduced (i, j, k)-patching operations. Since $9 \sim \log(8100)$, the expected value of the disjoint cycles is not greater than 9 under the problem size 8100. In practical point of view this limit of problem size is enough large, and so, our procedure uses (i, j, k)-patching operations while the number of the disjoint cycles is not greater than 9. For the extreme cases, when the number of the disjoint cycles is greater than 9, we apply an additional step (*Step 3*) to pair the small cycles with the large ones and to join them by suitable (i, j)-patching operations.

The 3-patching algorithm

Step 1. Determine an optimal assignment φ.

Step 2. If the current φ is a cyclic permutation then terminate. Otherwise go to Step 3.

Step 3. Let m denote the number of the cycles of φ. If $m \leq 9$ then go to Step 4. Otherwise order the cycles with respect to the number of vertices belonging to them. Let $U_1, \ldots, U_m$ denote the sequence of the cycles in increasing order. Calculate the 2-patching cost d_{rs} of the cycles U_r and U_{m-l+s} for all $1 \leq r \leq l$ and $1 \leq s \leq l$, where $l = [m/2]$. Solve the assignment problem of type $l \times l$ with the cost matrix $\mathbf{D} = (d_{rs})$. Let τ denote an optimal assignment. For all $1 \leq r \leq l$, join the cycles U_r and $U_{m-l+\tau(r)}$ with patching cost $d_{r\tau(r)}$, using a suitable (i, j)-patching operation. Consider the assignment obtained after the l joins as the current assignment φ and repeat Step 3.

Step 4. If $m = 2$, then determine the 2-patching cost of the two cycles of φ, join them with a suitable (i, j)-patching operation and terminate. If $m > 2$, then choose three cycles U, V, W of φ for which $\Theta(\varphi, U, V, W)$ is minimal. Determine three cities $i \in U$, $j \in V$, $k \in W$ with the 3-patching cost $\Theta(\varphi, U, V, W)$. Perform the (i, j, k)-patching operation and consider the new assignment as the current φ. If φ is a cyclic permutation then terminate. Otherwise repeat Step 4 with the current φ.

In order to efficiently find the three cycles having minimal 3-patching cost required in Step 4, we maintain a 3-dimensional array of the values $\Theta(\varphi, U, V, W)$. To set up this array we firstly need $O(n^3)$ operations, but during the iterations one can compute the new array from the previous one easily.

In both patching procedures above the starting point is an optimal assignment which can be computed by the Hungarian method in $O(n^3)$ steps. This method

starts with an independent set of zeroes in the reduced cost matrix. These independent zeroes can be determined randomly. In general, different independent sets of zeroes result in different optimal assignments. Using this observation and executing the procedure k times, we can obtain k distinct heuristic solutions and we choose the best of them.

To test the efficiency of our procedure we performed the following computational experiment on a 33MHz 486 machine. We randomly generated 100-100 problems under $n = 100$, $n = 150$, $n = 200$ $n = 250$, respectively, with costs independently drawn from a uniform distribution of the integers over the interval $[0, 100]$. We applied some classical tour construction heuristics and three versions of both the 2-patching and the 3- patching methods to obtain heuristic solutions for the generated problems. The difference among the three versions appears in the number k of executions under a randomly choosen independent set of zeroes. We worked with the values $k = 1, 3, 5$. Simultaneously, we solved these problems by a branch and bound procedure, using the best heuristic solution as the best known feasible solution. Regarding the classical heuristics we applied the all cities versions of the nearest addition, nearest insertion and farthest insertion algorithms, and our cheapest insertion procedure started from a shortest two city subtour. The results of our computational experiments are reported in Table 1. Here the first column gives the averages of the ratios $z(\textit{heuristic solution})/z(\textit{optimal solution})$, the second column contains the averages of the run-times in seconds and the third column shows how many times the suitable heuristic gave the best solution among the ones provided by all the 11 heuristics considered.

	$n = 100$			$n = 150$			$n = 200$			$n = 250$		
	average ratio	average sec.	best value	average ratio	average sec.	best value	average ratio	average sec.	best value	average ratio	average sec.	best value
B&B	1.000	40.78	-	1.000	87.69	-	1.000	194.1	-	1.000	320.9	-
3-patching k=5	1.054	19.53	88	1.056	58.55	81	1.052	190.2	82	1.059	370.8	80
3-patching k=3	1.061	11.60	70	1.063	34.93	67	1.061	122.0	54	1.078	218.6	48
3-patching k=1	1.069	3.84	55	1.096	11.90	39	1.094	39.8	25	1.134	72.6	24
2-patching k=5	1.090	11.14	33	1.082	29.70	38	1.069	88.4	40	1.101	158.3	37
2-patching k=3	1.092	6.69	28	1.097	17.92	26	1.085	53.1	27	1.119	94.8	23
2-patching k=1	1.108	2.21	21	1.127	6.04	15	1.127	17.7	13	1.177	31.5	12
cheapest insertion	4.654	7.77	0	6.794	37.48	0	9.934	89.7	0	18.11	175.4	0
nearest insertion	4.392	9.19	0	7.047	43.66	0	11.39	104.6	0	18.60	205.1	0
farthest insertion	4.534	8.76	0	7.110	43.71	0	11.39	104.6	0	18.60	205.3	0
nearest addition	18.43	7.09	0	33.84	32.72	0	57.51	77.0	0	98.43	149.7	0

Table 1

According to the obtained results, both patching algorithms appear to be better than the investigated insertion procedures. Moreover, the 3-patching method seems to provide a good approximate solution almost independently of the problem size. Due to the very good starting feasible solution, the branch and bound method also turns out to be rather effective.

Acknowledgment. The authors would like to thank R. E. Burkard for his valuable comments.

References

[1] P. C. Gilmore and R. E. Gomory, Sequencing a one state-variable machine: a solvable case of the traveling salesman problem, *Oper. Res.* **12**, 655-679 (1964).

[2] M. Hall, *Combinatorial Theory,* Blaisdell, Waltham, 1967.

[3] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1985

[4] L. Lovász, *Combinatorial Problems and Exercises*, Akadémiai Kiadó, Budapest, 1979.

[5] R. M. Karp, A patching algorithm for the nonsymmetric traveling salesman problem, *SIAM J. Comput.*, **8**, 561-575 (1979).

Received April, 1995