A Heuristic for the Resource-Constrained Traveling Salesman Problem

Gregory Gutin, Alexei Zverovitch
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

David Blokh
The Schottenstein Cellscan Center for Early Detection of Cancer
Department of Physics, Bar-Ilan University
Ramat Gan, 52900, Israel

1 Introduction

The Resource-Constrained Traveling Salesman Problem (RCTSP) is formulated as follows. A set of n cities is given (denoted by V); travelling from city i to city j incurs cost c_{ij} as well as resource consumption r_{ij}. The objective is to find a minimum cost Hamilton cycle (or *tour*) that consumes not more than R_{max} units of the resource, where R_{max} is some constant. The cost (resource consumption) of a tour is the sum of the costs (resource consumptions) of its arcs.

We study the asymmetric version of the problem, whereby generally $c_{ij} \neq c_{ji}$ and/or $r_{ij} \neq r_{ji}$ for at least some pairs (i, j).

RCTSP is a generalization of the well-known Traveling Salesman Problem (TSP); see, for example, [15] and [10]. RCTSP was first introduced by Pekny and Miller in [19], who have proposed an exact parallel branch-and-bound algorithm for solving the problem. They have also shown the Resource Constrained TSP to subsume the Prize Collecting TSP [4] and Orienteering Problem [8] as special cases. In [19], Pekny and Miller discuss an application of the RCTSP to scheduling a processing facility with sequence dependent transition costs and an aggregate deadline for job completion.

In this paper, we slightly modify and adapt to RCTSP a Lagrangian relaxation method for general NP-hard optimization problems that can be traced back to the 1970s. Since then, this method has been studied for special NP-hard problems in a large number of papers including [1, 2, 5, 7, 9, 11, 14]. A Lagrangian relaxation method is described in several monographs including [6, 18]. Recently, the authors of [14] managed to prove that the Lagrangian relaxation method is of strongly polynomial complexity for the restricted shortest path problem. A. Juttner (private communication, 2001) claims that the method used in [14] can be applied to prove strong polynomiality of the Lagrangian relaxation method for a quite general family of optimization problems.

We modify the Lagrangian relaxation scheme from [5]. The modification is required due to necessity to have certain parameters bounded and integral (for TSP solvers). Our computational experience shows that heuristic solutions are fairly close to optima. Thus, the Lagrangian relaxation method can be applied to RCTSP if a fast heuristic solution is required or the RCTSP data are not exact. Such heuristic solutions can be used as upper bounds for exact algorithms and as good initial solutions for metaheuristics.

2 Heuristic

Algorithm 1 presents a heuristic for RCTSP and, without the δ_c and δ_r scaling part, this is an adoption of the Lagrangian relaxation scheme from [5]. The scaling part and the rounding down for D is required since we need the entries of matrix (array) D to be within certain range.

Most of the TSP codes we are aware of are only capable of dealing with integer inter-city distances. Therefore when the matrix D is computed, its elements are rounded to an integer so that a TSP solver can be applied to D. This makes it necessary to scale δ_c and δ_r to minimize loss of precision as a result of rounding. The scaling is controlled by the parameter μ: the scaling results in one of the two coefficients (δ_c or δ_r) becoming equal to μ and the other becoming less than or equal to μ in absolute value.

In addition to minimizing precision loss, the parameter μ also bounds the values of δ_c and δ_r from above. This prevents uncontrollable growth of the coefficients and consequently the elements of matrix D.

Sometimes elements of D may become negative. If the TSP solver used is not able to deal with negative inter-city distances correctly, the following transformation can be applied to matrix D before executing the TSP solver: $d'_{ij} = d_{ij} - \min\{d_{ij} | 1 \leq i \leq n, 1 \leq j \leq n\}$, where n is the number of cities in the RCTSP.

The following notation is used Algorithm 1:

- μ is some constant used for scaling δ_c and δ_r; a typical value of μ on a 32-bit platform can be, e.g., 1000.
- $r(x)$ and $c(x)$ denote the cost of tour x w.r.t. matrices $R = [r_{ij}]$ and $C = [c_{ij}]$ respectively.
- $TSP(A)$ denotes a solution for the unconstrained TSP defined by distances matrix A. The algorithm does not require this to be an optimal solution, it can be an approximate solution obtained using some heuristic, for example, the Lin-Kernighan heuristic [16].

3 Lower Bound

In this section we derive a lower bound (LB) for the Resource-Constrained Traveling Salesman Problem.

Let $D_{\delta_r,\delta_c} := \lfloor \delta_r C + \delta_c R \rfloor$ for some δ_r and δ_c. Let

$$L(\delta_r, \delta_c) := \frac{1}{\delta_r}(\mathrm{LB}_{\mathrm{TSP}}(D_{\delta_r,\delta_c}) - \delta_c R_{max}),$$

where $\mathrm{LB}_{\mathrm{TSP}}(A)$ is a lower bound for the unconstrained TSP defined by matrix A. The following assertion is an adoption for RCTSP of the general Lagrangian lower bound, see, e.g., the papers (or monographs) on Lagrangian relaxation cited above.

Claim. For any $\delta_r > 0$ and $\delta_c \geq 0$, the value $L(\delta_r, \delta_c)$ provides a lower bound for the RCTSP defined by the cost matrix C and resource consumption matrix R with resource consumption constrained by R_{max}.

Proof: Assume f^* is an optimal solution to the RCTSP. Let $c(f^*)$ denote the cost and $r(f^*)$ the resource consumption of solution f^*.

$$\begin{aligned}
L(\delta_r, \delta_c) &= \frac{1}{\delta_r}(\mathrm{LB}_{\mathrm{TSP}}(D_{\delta_r,\delta_c}) - \delta_c R_{max}) \\
&\leq \frac{1}{\delta_r}(D_{\delta_r,\delta_c}(f^*) - \delta_c R_{max}) \\
&= \frac{1}{\delta_r}(\lfloor \delta_r c(f^*) + \delta_c r(f^*) \rfloor - \delta_c R_{max}) \\
&\leq c(f^*) + \frac{\delta_c}{\delta_r} r(f^*) - \frac{\delta_c}{\delta_r} R_{max} \\
&= c(f^*) + \frac{\delta_c}{\delta_r}(r(f^*) - R_{max}) \\
&\leq c(f^*) \quad \square
\end{aligned}$$

Algorithm 1 Heuristic for RCTSP

```
begin
    { STEP 1 - Optimize C }
    $x_c \leftarrow TSP(C)$
    if $r(x_c) <= R_{max}$ then
        {Optimal solution found}
        terminate and return $x_c$
    end if

    { STEP 2 - Optimize R }
    $x_r \leftarrow TSP(R)$

    $step \leftarrow 3$
    repeat
        { Compute $\delta_r$ and $\delta_c$ }
        $\delta_c \leftarrow c(x_r) - c(x_c)$
        $\delta_r \leftarrow r(x_c) - r(x_r)$

        { Scale $\delta_r$ and $\delta_c$ }
        if $|\delta_c| > |\delta_r|$ then
            $\delta_r \leftarrow \delta_r \mu / \delta_c$
            $\delta_c \leftarrow \mu$
        else
            $\delta_c \leftarrow \delta_c \mu / \delta_r$
            $\delta_r \leftarrow \mu$
        end if
        if this $(\delta_c, \delta_r)$ pair has already been tried before, then
            terminate and return $l_{max}$
        end if

        { Solve unconstrained TSP }
        $D \leftarrow \lfloor \delta_r C + \delta_c R \rfloor$
        $x \leftarrow TSP(D)$

        { Replace $x_c$ or $x_r$ }
        if $r(x) > R_{max}$ then $x_c \leftarrow x$ else $x_r \leftarrow x$ end if
        $step \leftarrow step + 1$
    end repeat
end
```

To obtain the best lower bound, one needs to maximize $L(\delta_r, \delta_c)$ over δ_r and δ_c:

$$L^* := \max_{\delta_r > 0, \delta_c \geq 0} L(\delta_r, \delta_c).$$

In order to approximate L^*, it is possible to use an iterative procedure similar to the one discussed in the previous section. The procedure is detailed in Algorithm 2. It uses the following notation:

- μ is some constant used for scaling δ_c and δ_r; a typical value of μ on a 32-bit platform can be, e.g., 1000.
- $r(l)$, $c(l)$, and $d(l)$ denote the cost of lower bound l w.r.t. matrices R, C, and D respectively.
- $LB_{TSP}(A)$ denotes a lower bound for the unconstrained TSP defined by distances matrix A.

This procedure relies on computing a lower bound for the unconstrained Traveling Salesman Problem. For example, the Held-Karp lower bound [12, 13] can be used.

4 Computational results

Following [19], we generate test RCTSP instances as follows. Cost matrix elements, c_{ij}, are independently and uniformly chosen random integers in the range $[0, 1000]$. (In this section, when we say "random", we in fact mean pseudorandom, generated using an implementation of the Mersenne Twister generator of [17]). Resource matrix elements, r_{ij}, are independently and uniformly chosen integers in the range $[0, 1000 - c_{ij}]$. The cost matrix $C = [c_{ij}]$ is generally not symmetric, since c_{ij} and c_{ji} are generated independently of each other and therefore may differ. This is also true for the resource matrix $R = [r_{ij}]$.

Maximum resource usage, R_{max}, is computed using the following formula:

$$R_{max} = \left\lceil \alpha \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ij}^c + (1 - \alpha) \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ij}^r \right\rceil \tag{1}$$

In the above formula, values x_{ij}^c represent an optimal solution of the unconstrained TSP defined by the cost matrix $[(c_{ij})]$; $x_{ij}^c = 1$ if the arc (i, j) belongs to the optimal TSP tour, $x_{ij}^c = 0$ otherwise. Similarly, x_{ij}^r represent

Algorithm 2 Lower bound for RCTSP

```
begin
    $l_c \leftarrow LB_{TSP}(C)$
    $l_r \leftarrow LB_{TSP}(R)$
    $l_{max} \leftarrow -\infty$
    repeat
        { Compute $\delta_r$ and $\delta_c$ }
        $\delta_c \leftarrow c(l_r) - c(l_c)$
        $\delta_r \leftarrow r(l_c) - r(l_r)$

        { Scale $\delta_r$ and $\delta_c$ }
        if $|\delta_c| > |\delta_r|$ then
            $\delta_r \leftarrow \delta_r \mu / \delta_c$
            $\delta_c \leftarrow \mu$
        else
            $\delta_c \leftarrow \delta_c \mu / \delta_r$
            $\delta_r \leftarrow \mu$
        end if
        if this $(\delta_c, \delta_r)$ pair has already been tried before, then
            terminate and return $l_{max}$
        end if

        { Compute lower bound }
        $D \leftarrow \lfloor \delta_r C + \delta_c R \rfloor$
        $t \leftarrow LB_{TSP}(D)$
        $l \leftarrow (d(t) - \delta_c R_{max}) / \delta_r$
        if $l > l_{max}$ then
            $l_{max} \leftarrow l$
        end if

        { Replace $l_c$ or $l_r$ }
        if $r(t) > R_{max}$ then
            $l_c \leftarrow t$
        else
            $l_r \leftarrow t$
        end if
    end repeat
end
```

an optimal solution of the unconstrained TSP defined by the resource matrix $[(r_{ij})]$.

In Equation (1), α is a parameter providing means for controlling tightness of the resource constraint. It assumes values between 0 and 1: when $\alpha = 0$, the resource constraint is very tight while still allowing for a feasible solution; when $\alpha = 1$, the resource constraint has no effect on the optimal solution of the problem since the problem is effectively reduced to the unconstrained TSP. We test the algorithm for the following values of α: 0.25, 0.5, 0.75 and 1.

We vary the size of the instances between 100 to 1000 cities, testing the algorithm on instances of the following sizes: 100, 316, and 1000 cities. For each value of α, we run the algorithm on ten different randomly generated instances of size 100, ten instances of size 316, and three instances of size 1000.

We use the `Concorde` code [3] to solve instances of the unconstrained Traveling Salesman Problem and to compute TSP lower bounds. Since `Concorde` supports symmetric TSP instances only, we first apply a transformation to convert an arbitrary TSP instance into a symmetric one. The transformation starts by splitting each city v_i into three, v_i^0, v_i^1 and v_i^2, and connecting v_i^0 to v_i^1 and v_i^1 to v_i^2 with undirected zero-cost edges. Each directed arc (v_j, v_k) from the original problem is then replaced by an undirected edge (v_j^2, v_k^0) of the same cost. Finally, the resulting graph is complemented to a complete undirected graph by adding all missing edges and setting their costs to some constant, large enough to ensure such edges are never selected as part of a "good" TSP tour.

The above transformation triples the size of the problem, and the largest problems in our testbed have their size increased by the transformation to 3000 cities. In order to be able to deal with such large problems we have modified the way R_{max} is computed. We use the same formula as above, (1), but substitute approximate TSP solutions instead of optima. Approximate solutions are computed using `Concorde`'s implementation of the Chained Lin-Kernighan algorithm which we run with default settings. We have used this method to compute R_{max} for all instances in our testbed.

We have implemented the RCTSP algorithm in the C++ programming language and used it to solve the instances of the testbed described above. The computational study was performed on an Intel Celeron 550MHz PC with 384 megabytes of RAM.

Table 1 shows the quality of the RCTSP solutions found by the algorithm. This is measured as a percentage excess of the cost of the tour found above the lower bound.

It can be seen from Table 1 that in all our tests the solutions found by the algorithm are within 25% above the lower bound. It is unclear how much of this gap is due to the approximate nature of the algorithm and how much is due to the lower bound not being very tight. One observation that can be made about the tightness of the lower bound is the following. When $\alpha = 1$, the heuristic always finds an optimal solution (this is due to the fact that the resource constraint is very loose when $\alpha = 1$). However, the gap between the (optimal) solution and the lower bound in some cases can be as high as 13.5%.

Table 2 shows the running time of the algorithm in seconds. For ease of implementation we run `Concorde` as a separate process. Each time a TSP instance needs to be solved, it is written to file in a format compatible with `Concorde`, and the `Concorde` executable is launched, which reads the file in and solves the TSP instance. This approach clearly results in some unnecessary overheads that can be avoided if the `Concorde` package is used as a callable library and embedded directly into the RCTSP application. Embedding `Concorde` into the RCTSP application will make the following operations unnecessary: 1) writing TSP instance to a file; 2) launching new process; 3) reading TSP instance from the file. We have measured the time required to perform these operations (as a function of the problem size) and computed estimated running time of the RCTSP algorithm if `Concorde` was used as a callable library rather than a separate executable. The estimated running times are presented in Table 3.

By comparing the data in Tables 2 and 3 it can be seen that the overheads dominate the running time of the algorithm on small problems ($n = 100$), accounting for up to 75% of the execution time of the algorithm. For medium-size problems ($n = 316$), the overheads account for a much smaller part of the running times, in the region of 15%. For large problems ($n = 1000$), the overheads take very little relative time, accounting for approximately 3-4% of the execution time.

5 Conclusion

In this paper we have studied a heuristic for the Resource-Constrained Traveling Salesman Problem (RCTSP), which is a modification of a Lagrangian relaxation scheme for certain NP-hard optimization problems. Our computational experiments show that the heuristic is very useful when a fast suboptimal solution is required. Fast suboptimal solutions can be used when computation time is restricted, as upper bounds for exact algorithms, and as

$\alpha = 0.25$				
	size	trials	avg excess	max excess
			over LB	over LB
	100	10	10.55%	22.54%
	316	10	13.90%	23.04%
	1000	3	13.28%	23.68%
$\alpha = 0.5$				
	size	trials	avg excess	max excess
			over LB	over LB
	100	10	6.74%	11.01%
	316	10	8.59%	17.65%
	1000	3	10.30%	14.37%
$\alpha = 0.75$				
	size	trials	avg excess	max excess
			over LB	over LB
	100	10	10.10%	19.69%
	316	10	8.41%	14.31%
	1000	3	9.36%	9.60%
$\alpha = 1$				
	size	trials	avg excess	max excess
			over LB	over LB
	100	10	3.59%	6.14%
	316	10	7.50%	9.70%
	1000	3	12.65%	13.53%

Table 1: Effectiveness of the algorithm

	size	trials	avg time, s	max time, s
$\alpha = 0.25$				
	size	trials	avg time, s	max time, s
	100	10	12.844	18.688
	316	10	118.805	217.140
	1000	3	1158.453	1601.579
$\alpha = 0.5$				
	size	trials	avg time, s	max time, s
	100	10	14.170	26.734
	316	10	119.149	271.547
	1000	3	1819.547	2451.828
$\alpha = 0.75$				
	size	trials	avg time, s	max time, s
	100	10	10.764	15.391
	316	10	151.059	263.656
	1000	3	1161.406	1388.079
$\alpha = 1$				
	size	trials	avg time, s	max time, s
	100	10	0.745	0.844
	316	10	4.224	4.453
	1000	3	42.063	43.954

Table 2: Algorithm running time

$\alpha = 0.25$				
	size	trials	avg time, s	max time, s
	100	10	3.122	4.509
	316	10	101.557	184.800
	1000	3	1111.341	1534.988
$\alpha = 0.5$				
	size	trials	avg time, s	max time, s
	100	10	3.822	8.366
	316	10	101.901	232.347
	1000	3	1744.349	2349.903
$\alpha = 0.75$				
	size	trials	avg time, s	max time, s
	100	10	2.790	4.196
	316	10	128.960	223.476
	1000	3	1113.841	1331.001
$\alpha = 1$				
	size	trials	avg time, s	max time, s
	100	10	0.297	0.396
	316	10	3.734	3.963
	1000	3	40.704	42.595

Table 3: Adjusted algorithm running time

good initial solutions for metaheuristics. It makes sense to restrict ourselves to suboptimal solutions also when the RCTSP data is not exact as it often happens in real world problems.

Acknowledgement

The research of GG has been partially supported by an EPSRC grant.

References

[1] V. Aggarwal, Y. Aneja, and K.P.K. Nair. Minimal spanning tree subject to a side constraint. *Comput. Operations Res.*, 9:287–296, 1982.

[2] Y.P. Aneja, V. Aggarwal, and K.P.K Nair. Shortest chain subject to side constraints. *Networks*, 13:295–302, 1983.

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde: A code for solving Traveling Salesman Problems. Available for download at `http://www.math.princeton.edu/tsp/concorde.html`.

[4] E. Balas. The Prize Collecting Traveling Salesman Problem and its applications. In Gutin and Punnen [10].

[5] D. Blokh and G. Gutin. An approximate algorithm for combinatorial optimization problems with two parameters. *Ausralasian J. Combin.*, 14:157–164, 1996.

[6] M.W. Carter and C.C. Price. *Oprations Research: A Practical Introduction.* CRC Press, Boca Raton, 2001.

[7] S.-J. Chung, S.-P. Hong, and H.-S. Huh. A fast algorithm for stereo matching. *Information Processing Letters*, 63:57–61, 1997.

[8] M. Fischetti, J.J. Salazar, and P. Toth. The generalized traveling salesman and orienteering problem. In Gutin and Punnen [10].

[9] D. Golenko-Ginzburg, D. Blokh, and G. Gutin. A two-parametric approximate method to optimize alternative activity network models. Part 1: The general approach and the algorithm. *Communications in Dependability and Quality Management*, 3:18–24, 2000.

[10] G. Gutin and A.P. Punnen, editors. *Traveling Salesman Problem and its Variations.* Kluwer, 2002.

[11] G.Y. Handler and I. Zang. A dual algorithm for constrained shortest path problem. *Networks*, 10:293–310, 1980.

[12] M. Held and R. Karp. The Traveling Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18:1138–1162, 1970.

[13] M. Held and R. Karp. The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.

[14] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko. Largrange relaxation based method for the QoS routing problem. In *IEEE INFOCOM 2001*, pages 100–109, 2001. Available for download at `http://www.ieee-infocom.org/2001/`.

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

[16] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[17] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[18] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, New York, 1988.

[19] J. F. Pekny and D.L. Miller. An Exact Parallel Alorithm for the Resource Constrained Traveling Salesman Problem with Application to Scheduling with an Aggregate Deadline. Technical Report EDRC 05-44-89, Institute for Complex Engineered Systems, Carnegie Mellon University, 1989.