A Linear Programming Based Iterative Heuristic for the Recreational Vehicle Scheduling Problem

S. Kulkarni[1,4,5], A. T. Ernst[2], A. Ranade[3], M. Krishnamoorthy[4]
[1]IITB-Monash Research Academy, Indian Institute of Technology Bombay, India
[2]School of Mathematical Sciences, Monash University, Australia
[3]Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India
[4]Department of Mechanical and Aerospace Engineering, Monash University, Australia
[5]S.J.M. School of Management, Indian Institute of Technology Bombay, India
[1]sarang.kulkarni@monash.edu

Abstract - **In the recreational vehicle rental business, the problem of preparing a schedule for each vehicle in the fleet (by assigning accepted bookings to available vehicles over the planning horizon) is known as the *recreational vehicle scheduling problem* (RVSP). The problem belongs to the class of minimum cost multicommodity network flow problems which is known to be NP-hard. A formulation of the RVSP from literature is modified to reduce the size of the problem instance. We propose an iterative construction heuristic combined with an improvement heuristic based on the solution of the LP relaxation of the problem. The heuristic exploits the integrality property of the formulation and reduces infeasibility successively at each iteration. The approach is compared with a subgradient optimisation and with CPLEX using a real-life dataset. The heuristic outperforms the subgradient optimisation for most of the datasets while it produces results within 5% of optimality when compared with CPLEX.**

Keywords – **heuristic, subgradient optimisation, vehicle scheduling, integer programming, assignment problem**

I. INTRODUCTION

Consider a scenario in which a recreational car rental company operates from different locations. The company operates with different types of vehicles. So at any point in time, different types of vehicles may be available at these locations and these may be used to satisfy customer bookings/requests. A booking is performed against a customer request that specifies when and where a vehicle or vehicle type needs to be picked up and dropped off. The pickup and drop off locations can be different; and in the case of recreational vehicles, they often are. A customer request also specifies a preferred vehicle type along with ranked list of acceptable substitutes, in case the requested vehicle type is unavailable. The bookings are made well in advance, sometimes a year earlier [1]. The task is to allocate vehicles to the accepted bookings over a known/given planning horizon in order to reduce various operational costs. From the context of a vehicle, it sequentially performs bookings assigned to it over the planning horizon.

Due to the nature of the recreational vehicle rental industry, large numbers of bookings are one-way (pickup and drop off locations are different). This creates an imbalance between supply and demand of the vehicles at different locations. A booking at any location can be satisfied if the requested vehicle type is available at the location in the required time period. If unavailable at the location, the vehicle of that type needs to be relocated from another location; this incurs a high relocation cost. Another option is to offer a different type of available vehicle to the customer. This is referred to as a substitution and incurs a substitution cost. The *recreational vehicle scheduling problem* (RVSP) attempts to satisfy all bookings while attaining a trade-off between operational costs, relocation costs and substitution costs.

The first article on the RVSP was by Kilby [1], in which the problem of maintaining feasible and near optimal schedule of vehicles in a real time environment was addressed.

A detailed description of the RVSP was presented by Ernst *et al.* [2], in which, various operational costs involved in the activity were discussed in detail. The problem was formulated as an integer program and ILOG CPLEX was used to minimize the operational costs.

Ernst *et al.* [3], presented two formulations for the RVSP: a network flow formulation and an assignment formulation. The network flow formulation was used to solve the static vehicle scheduling problem (which is addressed in the current study) while assignment formulation was used to address the real- time scheduling decisions. The network flow formulation was also used in making revenue management decisions.

Ernst *et al.* [4] presented a detailed description of the assignment formulation. They implemented a modification of the Wedelin algorithm [5] to propose a Lagrangean relaxation based heuristic for the RVSP.

As discussed above, there are two formulations used to model the RVSP. One approach is to model the RVSP as a minimum cost network flow problem on a time-expanded network. The other approach models the problem as different assignment models linked by a common set of constraints that prohibit assigning a booking to more than one vehicle type [4]. We used and modified the later formulation in our study. To reduce the size of the problem we used node aggregation wherever possible. The removal of a common set of constraints leads to independent assignment problems that satisfy the integrality property. Solving the LP relaxation of these independent problems provides an integer solution that may be infeasible due to the relaxed constraints. We propose an iterative approach that uses reduced cost

information from the solution of LPs at each iteration (to reduce the infeasibility for a set of bookings) thereby producing an integer feasible solution at the end of the procedure.

The rest of the paper is organised as follows. The assignment formulation with node aggregation is presented in Section II. The heuristic and solution methodology is presented in Section III. Section IV presents the results and discussion with conclusions and scope for future research provided in Section V.

II. PROBLEM FORMULATION

The assignment formulation proposed in the literature models the RVSP at the level of an individual vehicle. This requires a large number of connections (and hence variables), to represent the RVSP. For a detailed description of the assignment formulation, refer to Ernst *et al*. [4]. We modified the assignment formulation by aggregating nodes. This reduces the size of the problem instance. For vehicles that start at the same location and in the same time period, we use a one vehicle-start node, while bookings that end in the same time period and at the same location, we use a one booking-end node. By the definition of the assignment component, all vehicles of the same type and having the same end of service requirement are grouped into the one assignment component. Hence, all vehicle-end nodes can be combined into the one vehicle-end node. Thus, the resulting component is still a bipartite graph where the left hand side (LHS) partition contains all vehicle-start nodes and booking-end nodes, and the right hand side (RHS) partition contains a vehicle-end node and booking-start nodes. In the original assignment formulation, each node has a size of one, (flow through each node equals one). In the new formulation, the size of aggregated nodes equals the number of nodes that are aggregated to form the corresponding node. We use following notations to describe the new aggregated version of the assignment formulation and refer to it as Formulation-Z.

Notations:

$N =$ Set of assignment components,
$B^n =$ Set of bookings in the component n ∈N,
$V_s^n =$ Set of vehicle start nodes in the component n ∈N,
$B_e^n =$ Set of booking end nodes in the component n ∈N,
$B_s^n =$ Set of booking start nodes in the component n ∈N,
$v_e^n =$ End of service node for vehicles in n ∈N,
$b_s^n \in B_s^n =$ Node which is the start of booking $b \in B^n$,
$b_e^n \in B_e^n =$ Node which is the end of booking $b \in B^n$,
$N_b =$ Set of components in which booking b is present,
$L^n = V_s^n \cup B_e^n$ Set of all LHS partition nodes,
$R^n = v_e^n \cup B_s^n$ Set of all RHS partition nodes.

Parameters:

$a_{ij}^n = \begin{cases} 1, & \text{if node } i \text{ is connected to } j \text{ in component } n \\ 0, & \text{otherwise} \end{cases}$
$n_b =$ No. of components in which booking b is present
$p_l^n =$ size of node $l \in L^n$
$q_r^n =$ size of the node $r \in R^n$
$q_r^n = \begin{cases} \text{number of vehicles in the component n, if } r = v_e^n \\ 1, \quad \text{otherwise} \end{cases}$
$c_{ij}^n =$ Cost of sending unit flow from node i to node j

Variables:

$x_{ij}^n, where\ i \in L^n and\ j \in R^n$, depicts flow through arc (i, j) and for, $j \in B_s^n$, $x_{ij}^n \in \{0,1\}$

Mathematical formulation:

Minimise

$$\boldsymbol{Z = \sum_{n \in N} \sum_{i \in L_n} \sum_{j \in R_n} a_{ij}^n\, c_{ij}^n x_{ij}^n} \quad (1)$$

Subject to:

$$\sum_{i \in L_n} a_{ij}^n x_{in}^n = q_j^n, \forall j \in R_n, n \in N \quad (2)$$

$$\sum_{j \in R_n} a_{ij}^n x_{in}^n = p_i^n, \forall i \in L_n, n \in N \quad (3)$$

$$\sum_{n \in N_b} x_{b_e b_s}^n \geq n_b - 1, \ \forall b \in B \quad (4)$$

$$x_{ij}^n \geq 0 \ and \ integer \ \forall i \in L_n, j \in R_n, n \in N \quad (5)$$

The objective function minimises the total cost of allocation. Constraints (2) and (3) ensure that flow through each node equals the size of the node. Constraint (4), overbooking constraint, ensures that a booking will be allocated/satisfied in one component at the most. Relaxing constraint (4) produces a set of independent flow problems. It can be shown that the solution to the LP relaxation of these problems is an integer solution. We can add a source node and a sink node to each assignment component. The source node is connected to all nodes in the LHS partition with the capacity of arcs is equal to the size of the corresponding node. Similarly, each node in the RHS partition is connected to the sink node, with the capacity on connecting arcs is equal to the size of the corresponding node. The problem is then converted to a single commodity minimum cost maximum flow problem, which has integrality property due to its totally unimodular constraint coefficient matrix [6]. Thus, if we relax constraint (4) we need to solve the LP relaxation to obtain the integer solution. This solution may be infeasible for the original problem Z and a repair procedure needs to be employed to remove infeasibility. We propose a simple iterative procedure that removes infeasibility partially at each iteration and finally arrives at a feasible solution.

III. SOLUTION METHODOLOGY

A. *Construction heuristic*

Consider a problem Z_1, in which constraint (4) and the integrality requirement on the variables are relaxed. The problem Z_1 can be seen as a set of problems that can be solved independently. As stated in Section II, the solution to the LP relaxation of these problems will always provide integer solutions. The integer solution thus obtained may be infeasible for the original problem because bookings may get allocated in more than one (independent) component.

We propose an iterative heuristic that partially removes infeasibility at each iteration by deciding the most desirable component for each over-allocated booking. We start the iterations by keeping the bookings into the corresponding components in which they have been allocated in the LP solution. At each iteration, we solve the problem Z_1. The reduced cost associated with variable $x^n_{b_e b_s}$, in the assignment component, $n \in N$, in which booking, b, was allocated, in the solution of Z_1 will provide information about the increase in the objective value if the booking is not performed in the respective component. Hence, we retain a booking b in only that component in which variable $x^n_{b_e b_s}$ has the highest reduced cost. Ties are broken arbitrarily, and modified Z_1 is solved again. At each iteration, we remove infeasibilities that may be caused due to the sharing of some of the bookings. As the number of bookings is finite, this procedure will always progress towards the feasible solution. If the gap between the current feasible solution and the solution of the LP relaxation of problem Z is within 1%, we stop; else, we use an improvement heuristic, which is discussed in the next section, to improve the solution.

B. Improvement heuristic

For small-to-medium sized problem instances, the construction heuristic produces good solutions fairly quickly. For larger instances, the construction heuristic struggles to find the good quality solutions. Hence, we propose an improvement heuristic to improve the solution that is obtained by the construction heuristic. The improvement heuristic uses the information from the solution of the LP relaxation of Z to guide the allocations of bookings to components. The infeasible solution of the LP relaxation may have some bookings allocated in exactly one component while other bookings will have allocations in more than one components. Let, S be the set of bookings that has received fractional allocation from more than one component in the LP solution and P^i, $i \in S$ be the set of components in which booking $i \in S$ is fractionally allocated in the LP solution of problem Z.

We start with the solution produced by the construction heuristic. Let $C[i]$ denote the component in which booking is allocated in the LP solution of the problem Z. Then, for each booking $i \in S$, in turn, we consider alternate choices for each $C[i]$ from the set P^i.

Specifically, the improvement heuristic maintains the current mapping, A, of bookings to components. A is initialised to C. Let $f(A)$ denote the cost of the solution in which booking i is placed in the component $A[i]$. For each i, which is taken in the order described below, the heuristic runs $|P^i|$ iterations. In the k^{th} iteration, $A[i]$ is set to the k^{th} element of the P^i. If the cost $f(A)$ of the new A improves, we accept the new A as the current best solution. At the end of $|P^i|$ iterations, booking, i, will be placed in one of the components from P^i, or it may continue to remain in the component $C[i]$. Note that $C[i]$ might be 'empty', if the solution produced by the improvement heuristic did not allocate booking, i, at all.

We consider bookings in decreasing order of y_i, where y_i denotes the dual value that is associated with booking i for inequality (4). A higher value of y_i indicates the tightness of the constraint and hence, we believe, it must be resolved earlier by the improvement heuristic. To find $f(A)$, we simply fix $x^n_{b_e b_s}$, as per A and then solve the linear program Z_1. As argued above, this is guaranteed to give an integer solution.

At any iteration, we make changes to two components at the most, which is enough to solve the LP for these components in order to reach the new solution. The removal and addition of booking, b, to component, n, is achieved by changing the bounds of the non-performing arc variables, $x^n_{b_e b_s}$. We prohibit a booking, b, from getting an allocation in component n by setting the lower and upper bounds for the non-performing arc variable, $x^n_{b_e b_s}$, to one. By restoring the bounds to zero and one, respectively, we can introduce booking, b, in component, n. The procedure eliminates the need to reformulate the model at each iteration. The flow chart for the complete heuristic is depicted in Fig. 1.

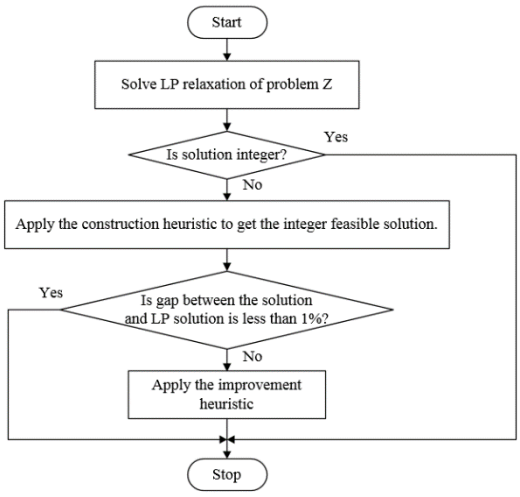

Fig. 1. Flow chart of the heuristic

C. Subgradient optimisation

We compared the performance of our heuristic with a subgradient optimisation procedure. By dualising constraints (4) with the help of Lagrangean multipliers $\lambda_b, b \in B$, we get the Lagrangean lower bound problem (LLBP).

$$Z_{LLBP}(\lambda_b) = \min \sum_{n \in N} \sum_{i \in L_n} \sum_{j \in R_n} a^n_{ij}\, c^n_{ij} x^n_{ij} + \sum_{b \in B} \lambda_b \left(n_b - 1 - \sum_{n \in N} x^n_{b_e b_s}\right) \quad (6)$$

Subject to:

Constraints (2), (3)

$$\boldsymbol{x^n_{ij} \geq 0 \;\; \forall i \in L_n, j \in R_n, n \in N} \quad (7)$$

For a given set of multipliers, $\lambda_b, b \in B$, Z_{LLBP} provides a lower bound on the original problem Z. We are interested in finding the multipliers that provide the maximum lower bound. Subgradient optimisation is an iterative procedure to update the multipliers and to provide a better lower bound to the original problem. At each iteration, the solution to LLBP may have infeasibility induced due to the over-allocation of some of the bookings. We used the construction heuristic described above to regain the feasibility and to update the upper bound of the problem at each iteration of the subgradient procedure. We have used the implementation of the subgradient method, presented by Smith and Thompson [7], with some of the improvements suggested by Beasley [8].

IV. RESULTS AND DISCUSSIONS

Twenty real-life problem instances have been used to test the approaches that are described in the paper. The instances vary in the number of bookings, components and vehicles. The problem instances are numbered in the ascending order of the number of connections in the aggregated assignment formulation. The number of connections is the same as the number of variables in the formulation, and this is a good indicator of the size of the problem.

Table I presents the description of the dataset. The column with heading 'Connections', provides a measure of size of the instance. The number of connections is equal to the number of variables. Hence, larger the number of connections, the bigger the instance size.

We implemented all the approaches using C++. To solve linear programs and integer programs, we used the CPLEX solver version 12.6 using Concert technology on a machine with CPU specifications described in Table II.

To ensure better readability in the figure, we term the assignment formulation without aggregation as (A1), and the assignment formulation with aggregation of nodes as (A2). Fig. 2 shows a comparison, between A1 and A2, of the number of connections and the CPU time required to solve the RVSP, for first ten instances. It is seen that the number of connections (and hence, the size of the problem) is reduced due to the aggregation of nodes. This results in a smaller solution time.

TABLE I
DATASET DESCRIPTION

Problem instances	Number of			
	Components	Vehicles	Bookings	Connections
1	22	1768	5	30,245
2	15	1181	11	68,088
3	22	1819	5	72,271
4	12	631	3	2,16,213
5	11	549	3	2,59,772
6	9	1043	10	4,33,550
7	10	1101	10	5,19,211
8	75	1308	14	8,28,272
9	17	1255	10	9,26,131
10	23	1236	10	11,49,128
11	93	2049	5	29,41,833
12	93	1765	7	33,41,393
13	99	2138	6	35,18,575
14	102	2275	6	35,53,585
15	109	1997	6	39,01,737
16	16	1366	11	43,98,391
17	94	2062	6	59,86,328
18	36	1955	13	63,73,399
19	35	2054	5	78,96,238
20	33	2052	5	79,05,038

We compared the subgradient approach with the heuristic approach, using the first ten instances. Table III shows the comparison of solution time and solution value between the heuristic and subgradient approach. It is observed that the heuristic approach performed better in terms of solution quality and solution time for most of the instances. We note that instance number five and eight are the only instances, among the ten, in which the solution of the LP relaxation is fractional and hence the heuristic is utilized to get the feasible integer solution. For both the instances, the heuristic approach outperformed subgradient optimisation.

Since the heuristic approach outperformed the subgradient method, we compared the heuristic approach with the CPLEX solution, for the next ten larger instances. Fig. 3 and Fig. 4 show the comparison of the solution time (and the value) between CPLEX and the heuristic.

TABLE II
CPU SPECIFICATION

Model	Intel(R) Xeon (R) CPU E5-2670 v2 @2.50GHz
RAM	128 GB
CPUs	20
Threads per core	1
Core per socket	10
OS	Linux 64bit

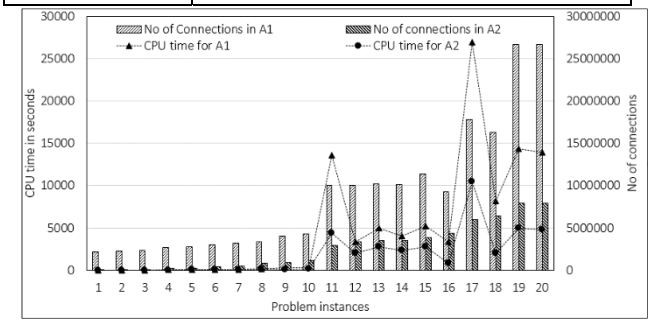

Fig. 2 Improvement due to node aggregation

TABLE III
COMPARISON BETWEEN HEURISTIC AND SUBGRADIENT SOLUTION

Problem instances	CPU time in seconds		% Deviation from the CPLEX solution	
	Subgradient	Heuristic	Subgradient	Heuristic
1	11.92	0.08	0.00	0.00
2	162.04	0.12	0.00	0.00
3	411.03	0.16	0.00	0.00
4	114.73	0.25	0.00	0.00
5	252.88	1.22	2.72	0.00
6	381.30	1.02	0.00	0.00
7	1125.89	1.39	0.04	0.00
8	26951.00	5.50	20.07	0.00
9	4657.31	4.53	1.73	0.00
10	3883.80	6.46	2.24	0.00

For all the instances, the solution time for the heuristic is lesser than the CPLEX solution time, while the maximum gap between the solutions of the heuristic and the LP solution is 4.5%. For instance 15 where CPLEX solution is 5% away from the LP relaxation, the heuristic is able to find a solution that is within 1% of the LP solution value.

Fig. 5 shows the division of the solution time for the heuristic into the percentage of the total solution time that is required (a) to solve LP relaxation, (b) the construction heuristic and (c) improvement heuristic, (for the instances where the heuristic is used to improve the solution).

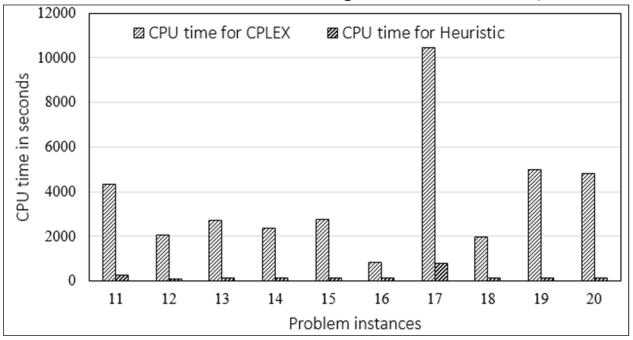

Fig. 3. Comparison of CPU time for CPLEX and heuristic

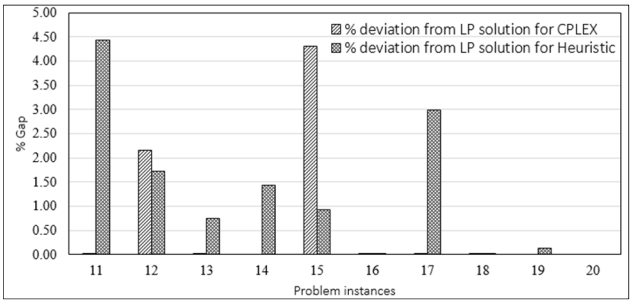

Fig. 4. Comparison of solution quality between CPLEX and heuristic

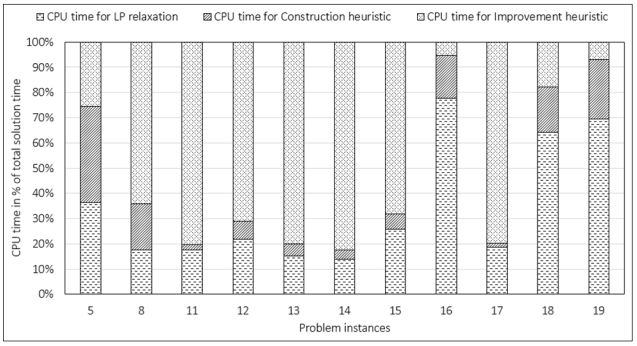

Fig. 5. Percentage of CPU time utilized between LP relaxation, construction heuristic and improvement heuristic.

It is observed that, with the increase in the instance size, the solution to the LP relaxation requires more time. The large amount of time that is required for the improvement heuristic implies that a greater number bookings are fractionally allocated to multiple components in the LP solution.

V. CONCLUSION AND FUTURE SCOPE

The proposed heuristic outperforms subgradient optimisation in terms of the solution quality as well as the solution time. When applied to large-sized instances, the heuristic provides solutions that are within 5% of the lower bound provided by the LP solution. The solution time for the heuristic is small, in comparison with the CPLEX solution time.

The major limitation of the heuristic is that it needs to start with an LP solution. As the instance size increases, getting the LP solution itself requires more computational time. Hence, a better starting solution technique needs to be developed to reduce the solution time further.

The techniques such as subgradient optimisation, that generate different solutions, at each iteration, can be used to generate different starting solutions which will be repaired and improved using the heuristic.

An interesting extension of this study could be the application of the heuristic approach to other multicommodity network flow problems, in which resources or commodities are assigned to multiple tasks and the tasks can be performed by more than one commodity or resource. In such cases, predetermining the resource-task allocation can reduce the problem into independent easy-to-solve problems which enables the derivation of upper bounds.

ACKNOWLEDGMENT

We thank the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia for providing the datasets for the experimentation. We also thank Prof. Rahul Patil, S.J.M. School of Management, IIT Bombay, for his valuable suggestions.

REFERENCES

[1] P. Kilby, "An online schedule update algorithm for vehicle fleets CMIS technical report number 04/17" 2004

[2] A. T. Ernst, M. Horn, M. Krishnamoorthy, P. Kilby, P. Degenhardt, M. Moran, “Static and dynamic order scheduling or recreational rental vehicles at tourism holdings limited” Interfaces. vol. 37, no 4, pp. 334-41, 2007

[3] A. T. Ernst, M. Horn, P. Kilby, M. Krishnamoorthy, “Dynamic scheduling of recreational rental vehicles with revenue management extensions”. Journal of the Operational Research Society. vol. 61, no. 7, pp. 1133-43, 2010

[4] A. T. Ernst, E. O. Gavriliouk, L. Marquez. “An efficient Lagrangean heuristic for rental vehicle scheduling.” Computers and Operations Research. vol. 38, no. 1, pp. 216-26, 2011

[5] Wedelin D., “An algorithm for large scale 0-1 integer programming with application to airline crew scheduling” European Journal of Operations Research, vol. 87, pp. 722-30, 1998

[6] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, “Network Flows: Theory, Algorithms, and Applications,” Prentice-Hall, 1993, ch. 11, pp 447-449.

[7] T. H. Smith, G. L. Thompson, “A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation,” Annals of Discrete Mathematics, vol. 1, pp 479-493, 1977.

[8] J. E. Beasley, “Lagrangean relaxation,” in Modern heuristic techniques for combinatorial problems: John Wiley & Sons, Inc, 1993, pp. 243-303.