A Meta-heuristic Applied for a Topologic Pickup and Delivery Problem with Time Windows Constraints

Jesús Fabián López Pérez

Post-Graduate Program of Management Science, FACPYA UANL, Monterrey, México
fabian.lopez@e-arca.com.mx

Abstract. Our problem is about a routing of a vehicle with product pickup and delivery and with time window constraints. This problem requires to be attended with instances of medium scale $(\text{nodes} \geq 100)$. A strong active time window exists $(\geq 90\%)$ with a large factor of amplitude $(\geq 75\%)$. This problem is NP-hard and for such motive the application of an exact method is limited by the computational time. This paper proposes a specialized genetic algorithm. We report good solutions in computational times below 5 minutes.

Keywords: Logistics, Genetic Algorithms, NP-Hard, Time Windows.

1 Problem Definition and Bibliographical Review

The objective is to determine the optimal route for a distribution vehicle. The vehicle departs from a distribution center and returns to the same point at the end of the route. An optimal route is defined as that which visits all the clients in such a way that we incur a minimal cost. We define a cost matrix which identifies the time or distance required to go from each client to all others. The problem constraints are as follows.

a. Each client visited has a requirement of product to be delivered and a load to be collected. We have to observe a finite load capacity for the vehicle all the time.
b. The time window identified for each client is defined by an opening hour and a closing hour. The time window width is equal to the difference between the closing hour and the opening hour. The visit to each client must be within the time window. It is not permitted to arrive before the opening hour nor after the closing hour.

Our bibliographical summary of previous investigations include Applegate et al. 1998 [1]; Dumas & Solomon 1995 [2]; Eijl Van 1995 [3]. The outlined problem is combinatoric in nature and is catalogued as NP-Hard, Tsitsiklis 1992 [4]. Regarding routing application area, the less investigated variant is the one which has to do with the physical product distribution, Mitrovic 1998 [5]. The instances that have been typically tested are characterized by time windows with a low percentage of overlapping, Ascheuer et al. 2001 [6]. The computational complexity for the solution of the SPDP-TW depends strongly on the structure of the time windows that are defined for each customer. The experimental results obtained by Ascheuer et al. proved that the TSP-TW is particularly difficult to be solved for instances with

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 924 – 928, 2005.
© Springer-Verlag Berlin Heidelberg 2005

more than 50% of active nodes with time window constraints. Ascheuer, Jünger & Reinelt 2000 [7] worked with instances up to 233 nodes. They reported 5.95 minutes of computational time for an instance of 69 nodes. All the greater instances required more than 5 hours of computational time. They conclude that the instances on the limit up to 70 nodes can be solved to optimality by exact methods like the Branch & Cut algorithms (B&C).

2 Methodology Proposed

Our methodology proposes 6 routines. We have 4 preprocessing routines, the Genetic Algorithm and finally one another routine for post-processing. We expose the 6 phases:

1. **Network topology decomposition phase based on a *"shortest path algorithm (SPP)"*.** We consider here the topology corners (street corners) that are required to model the traffic constraints that we have to observe in order to arrive to each customer in the network. This pre-processing strategy contributes to reduce the computational complexity during all the subsequent phases. In order to simplify our formulation we define an empirical assumption. We are going to use a constant "4" as the quantity of network arcs (north, south, east & west) that we require to model a typical city street corner. With the previous assumption we can establish that, if we setup a network with N1 nodes, we would obtain only N2 nodes, where $N1 \approx 4N2$. We use an SPP algorithm to pre-calculate the optimal sub-tour required to move from each customer to each one of the rest. All these preprocessed sub-tours fill the *N2* cost matrix to be used in the next phases.
2. **Compressing & clustering phase through a *"neighborhood heuristic"*.** The *N2* nodes are grouped to setup a reduced quantity of *N3* meta-nodes *(where: N3 < N2)*. Our assumption here is that we have driving times near to zero between the nodes that are going to be grouped. This assumption is going to be considered valid through all the phases where we work with the reduced version of the network. Taking in mind this, we can inference that the obtained solution for the reduced network can be derived as an optimal for the original network as well. The heuristic that we use here to group the nodes is by geographically neighborhood and also by time windows structure similarity. Starting from a group of nodes to be grouped in a meta-node, the time window structure of this meta-node is defined by the latest opening time and by the earliest closing time. We use in the algorithm a 50% compression factor for the grouping phase which means that $N2=2*N3$.
3. **Discriminate compressing phase through a *"k nearest nodes heuristic"*.** The network arcs with greater cost are eliminated from the matrix. The logic of the previous assumption is because of those arcs have a smaller probability to appear in the optimal solution. For each *N3* node in the network, we maintain only the *"k"* arcs with the smallest cost, *where k<<N3*. We use a conservative 20% discriminate factor in order to reduce the probability that the optimal solution go out from the search space. This empirical assumption means that the matrix that will be transferred to the next phase will be reduced and defined by *N3 x N4, where N4 = 20% * N3* in an "incidence sense". This means that although the dimensionality of the matrix is still the same *(N3 x N3)*, the quantity of non-zero elements in the matrix is reduced to an equivalent matrix size of *N3 x N4*.
4. **Aggressive Branch & Cut phase.** the initial math formulation we use here is quite similar as we may found in a basic TSP problem [6]. We have that X_{ij} formulation means the existence of an arc from *"i" to "j"* in the route. The procedure to add sub-tour elimination constraints are also included on this phase. Starting with *N3 meta-nodes*, the objective is to find as quickly as possible, the first feasible solution that cover the time window and

vehicle capacity constraints. The logic that we apply here is to iteratively generate cuts within a Branch & Cut scheme. For that purpose we identify in the incumbent solution, the node with the greater deviation in relation to the time window and/or the vehicle capacity constraint. This node is named *"pivot node"*. Then we verify the nodes of the tour that can be identified as *"related"* in order to re-sequence the position of the *pivot node* within the tour. This relation of the *pivot node* is exploded in the generation of the cut. The logic that we apply here to generate the cut assures that the *pivot node "k"* use at least one of the arcs that connect it to one of the *related nodes "j"*. This procedure continues until is found the first feasible solution. At this stage we use Xpress Ver 15.10 ©

$$\exists\ I = \{1..N_3\}\ \textbf{(network nodes)}$$
$$k \in I\ \textbf{(pivot node)}$$
$$j \subseteq I\ \{1..m\}\textbf{(related nodes to k)}$$
$$\sum_{j=1}^{m} (x_{jk} + x_{kj}) \geq 1 \quad \forall\, k \subseteq I$$

5. **Evolutionary phase.** our objective here is to approximate the optimal solution for the compact version of the network. Maintain in the pool of constraints a cut unnecessarily, means to take out the optimal solution or at least a better solution, from the search space. Our computational experience indicates that the quantity of cuts that get to be accumulated in the pool is meaningful (15-40 cuts). The goal is to identify which cuts of the pool are necessary to be eliminated. The cut elimination procedure can not be seen as an individual process for each cut, since the presence and/or the elimination of any cut can commit simultaneously the presence and/or the elimination of other(s). Identify which cuts must be eliminated, can be seen as a combinatoric sub-problem. We then propose an evolutionary strategy to attend this sub-problem. A binary codification permits to represent the elimination (0) and the presence (1) of a cut in the pool. Our Genetic Algorithm applies a tournament selection operator with a 50% crossing factor. The reproduction method applied was by means of a two random crossing points throughout the chromosome length. The mutation factor is initialized with a 5% value and it is auto-adjusted in each generation depending on the percentage of individuals in the population with identical genetic material content. Upon increasing the degeneracy level in the population, is applied an exponential growth curve in the mutation factor with 50% as an asymptotic limit. The elitism factor is limited to 15% of the population. The fitness function we calculate in this evolutionary stage is related with two objectives. The first is the route cost minimization. The second is the infeasibility level we calculate in the route as we may delete some cut(s) in the chromosome. The infeasibility level is only related with the capacity and time windows constraints for the SPDP-TW formulation. Our objective on this stage is to find the subset of cuts that can be deleted from the pool and at the same time we obtain a feasible solution. The final solution at this stage is when we found a minimal cost route which is still feasible as well. We remark here that our methodology is different from the conventional approach in the evolutionary bibliography. Our review indicates that most of the evolutionary approaches to attend routing problems take in mind the genes as the nodes or sequence nodes in the route. We treat with a modified problem. The genes represent the presence or elimination of a cut in the math formulation of the problem that is actually modeling the route.
6. **Uncompressing phase to generate a route for the original network.** The post-processing phase has the objective of translating the solution obtained in the compact version of the network to one another that will be topology sense equivalent to the original network. We have here 2 routines. The first routine is focused in determining the optimal sequence on which the *N3 meta-nodes* should be disaggregated to return to the *N2 nodes* obtained in phase 2. Starting from a selected meta-node, we construct only the valid arcs to the previous and to the next meta-node. This procedure is propagated to the rest of the

meta-nodes in the network. The second conversion routine makes use of the topology information generated in the first phase of our general methodology. Its objective is to substitute the sequence of the tour defined by the N2 nodes according to the cardinal movements that are required to obtain the N1 nodes that are present on the original network.

3 Experimental Development and Results

We applied an "*Experimental Design"* through the use of 4 experimental instruments: (1) B&C Algorithm (Xpress© Ver 15.10); (2) Steady Sate Genetic Algorithm (Evolver© Ver 6.0); (3) Generational Genetic Algorithm (Solver© Ver 4.0); (4) Proposed Genetic Algorithm. We will calculate a "percentage of optimality" for the statistic test of the hypothesis.

$$optimality\ \% = 1 - \frac{GA\ solution - Lower\ Bound}{Lower\ Bound}$$

$$where\ Lower\ Bound = Best\ solution\ reached\ by\ B\&C\ within\ 5\ hours\ limit$$

We gave treatment to instances with more than 70% of active time windows and with a minimal width of 75%. The dimension of the tested instances are defined by *w,* where $(100 \leq w \leq 120)$. The genetic parameters applied for the implementation of each GA's # 1, 2 & 3 were adapted empirically and separately to different values accordingly to the best case scenario. That means that the parameters were tuned for each GA. The experimental design was applied for a sample of 40 instances. All these instances were randomly generated. Only the B&C instrument was limited up to 5 hours of computational time. We remark here that although 5 hours of computational time is not evidence of optimality, we can report that we obtain the optimal solution in 38 of 40 instances. For the previous described GA instruments, the "% of optimality" was applied in 4 successive moments of time (minute #3, #5, #8 and #10). We define the following statistic parameters: *(1) Mean (m) & (2) Standard deviation (s).* The T Student test "*P(x> 90%)"* applied for each element (m_{ij}, s_{ij}) calculates the probability that the *algorithmic instrument "j"* in the *time interval "i"* obtains at least a 90% of optimality. Table 1 shows the values calculated for the "T" statistic. Table 2 shows the probability coefficients "P" Value.

Table 1. "T" Statistic Values

	Algorithmic instruments to be compared					
	B&C Algorithm (Control Group)	Basic Genetic Algorithm (Evolver)	Basic Genetic Algorithm (Frontline)	Proposed Genetic Algorithm		
				P(x>90%)	P(x>92.5%)	P(x>95%)
3th Minute	NA	-0.404	-0.558	2.426	1.069	-0.091
5th Minute	-2.426	-0.116	-0.307	3.313	1.539	0.111
8th Minute	-1.280	0.162	0.317	4.851	4.105	0.830
10th Minute	-0.700	0.400	0.903	6.298	5.577	1.328

Table 2. Probability "P Values"

	Algorithmic instruments to be compared					
	B&C Algorithm (Control Group)	Basic Genetic Algorithm (Evolver)	Basic Genetic Algorithm (Frontline)	Proposed Genetic Algorithm		
				P(x>90%)	P(x>92.5%)	P(x>95%)
3th Minute	NA	34%	29%	99%	85%	46%
5th Minute	<1%	45%	38%	100%	93%	54%
8th Minute	10%	56%	62%	100%	100%	79%
10th Minute	24%	65%	81%	100%	100%	90%

4 Discussion and Conclusions

Our B&C implementation obtains the optimal solution for 38 of 40 instances that are particularly difficult to be solved and where the investigation is focused. In addition we tested some "toy" instances with lees than 70 nodes and with less than 60% of active time windows. The computational times were very favorable since we report times below 3 minutes. Our proposed GA #3 obtains satisfactory solutions ($\textbf{\textit{optimality}} \geq 90\%$) and in reasonable computational times $\textbf{\textit{(3} \leq t \leq \textbf{5 minutes)}}$. Both "out of the self" GA's (1 & 2), are significantly inferiors since these never surpass 90% of optimality before the fifth minute. We conclude:

1. *We can establish that the proposed methodology reaches a percentage of optimality* $\geq$ *90% in a computational time* $\geq$ *5 minutes (0.001 significance level).*
2. Table 1 & 2 shows that the proposed GA offers solutions within an acceptable optimality range and with computational times that make feasible its implementation. However, we should establish that our methodology can assure only 54% of confidence when is required to reach an optimality $\geq 95\%$ in a computational time $\geq$ 5 minutes.

References

[1] Applegate, D; Bixby, R; Chvátal, V; (1998), On the solution of traveling salesman problems. "Documenta Mathematica Extra Volume ICM III", USA.
[2] Dumas, Y; Desrosiers, J; Solomon, M. (1995), An algorithm for the traveling salesman problem with time windows, "Operations Research 43(2)", USA, pp. 23-25.
[3] Eijl Van, C. (1995), A polyhedral approach to the delivery man problem, "Tech Report 95–19", Dep of Maths and Computer Science, Eindhoven Univ of Technology, the Netherlands.
[4] Tsitsiklis, J. (1992), Special cases of traveling salesman and repairman problems with time windows, "Networks No. 22", USA.
[5] Mitrovic, Snezana. (1998), Pickup and Delivery Problem with Time Windows, "Technical Report SFU CMPT TR 1998-12", Canada, pp 38-39.
[6] Ascheuer, N; Fischetti, M; Grotschel, M. (2001), Solving ATSP with time windows by branch-and-cut, Springer-Verlag, Germany.
[7] Ascheuer, N; Jünger, M; Reinelt, G. (2000), A branch & cut algorithm for the ATSP with precedence constraints, "Comput. Optimization and Applications 17(1)", USA, pp. 2-7.