A HYBRID CONSTRAINT PROGRAMMING / LOCAL SEARCH APPROACH TO THE PICK-UP AND DELIVERY PROBLEM WITH TIME WINDOWS

Daniel Guimarans[(a)], Juan José Ramos[(b)], Mark Wallace[(c)] , Daniel Riera[(d)]

[(a)(b)]Telecommunication and System Engineering Dept., Universitat Autònoma de Barcelona, Spain
[(c)]Faculty of Information Technology, Monash University, Melbourne, Australia
[(d)]Estudis d'Informàtica, Multimèdia i Telecomunicació, Universitat Oberta de Catalunya, Barcelona, Spain

[(a)]Daniel.Guimarans@uab.cat, [(b)]JuanJose.Ramos@uab.cat, [(c)]Mark.Wallace@infotech.monash.edu.au, [(d)]drierat@uoc.edu

ABSTRACT
Routing vehicles to serve customers is a problem that naturally arises in many distribution systems. Moreover, fleet management requires fast algorithms able to cope with continuously changing needs. Many efforts have been addressed to tackle different vehicle routing problem's variants. Among them, the pick up and delivery problem with time windows (PDTW) has received far less attention despite its relevance from practical and theoretical perspectives. The present paper provides a hybrid approach to the PDTW based on Constraint Programming paradigm and local search. Indeed, the proposed algorithm includes some performance improvements to enhance its efficiency. Thus, this hybrid approach may provide a solution to problems otherwise intractable in a reasonable computational time, as shown in the presented results. Due to these characteristics, the proposed algorithm may be an efficient tool in decision making support, as well as a mechanism able to provide an initial solution for subsequent optimization techniques.

Keywords: pick up and delivery problem, hybrid algorithm, constraint programming, local search

1. INTRODUCTION
An important component of many distribution systems is routing vehicles to serve customers. In fact, many companies are faced with problems regarding the transportation of people, goods or information. In many cases, they have to efficiently manage a heterogeneous vehicle fleet providing pick-up and delivery services to a set of customers. These companies have to optimize transportation by using rational manners and effective tools.

This class of logistics problems is usually known as the vehicle routing problem (VRP) and its objective is usually twofold: to minimise travelling costs by arranging visits in a proper sequence, while reducing the number of vehicles to be used in order to serve all customers. In the field of combinatorial optimization, the VRP is regarded as one of the most challenging problems because of its NP-Hardness (Savelsbergh 1985). For such problems in real situations, it is often desirable to obtain approximate solutions, so they can be found fast enough and are sufficiently accurate for the purpose.

The most basic VRP is the capacitated vehicle routing problem (CVRP) that assumes a fleet of vehicles of fixed capacity housed in a central depot. More realistic routing problems include travel times between every pair of nodes, customer service times and the maximum tour duration as additional problem data. These characteristics define the VRP with time windows (VRPTW) as a generalization of the CVRP. The VRP with pick-up and delivery and time windows, or pick-up and delivery problem with time windows (PDTW), is a generalization of the VRPTW. In the PDTW, pairs are defined among customers so pick up and drop off locations are determined, in addition to time windows constraints related to each visit.

Several classes of VRP have been studied in the literature. The VRPTW has been the subject of intensive research efforts for both heuristic and exact optimization approaches. Because of the high complexity level of the VRPTW and its wide applicability to real situations, solution techniques capable of producing good solutions in limited time, i.e. heuristics, are of major importance (Bräysy and Gendreau 2005a). Over last few years, many new heuristic approaches have been proposed, primarily metaheuristics, for tackling the VRPTW (Bräysy and Gendreau 2005b). With respect to the literature on VRPs, it is interesting to observe that although PDTW is as important and interesting from practical and theoretical points of view, it has received far less attention (Lau and Liang 2001; Savelsbergh and Sol 1995).

The present paper is structured as follow: section 2 introduces the proposed PDTW model, where the complete problem is decomposed into two separated subproblems. Next section includes a description of the adopted hybrid approach and algorithm's general structure, as well as some performance improvements added to the model. Section 4 presents some results obtained by using the proposed algorithm and corresponding analysis. Finally, conclusions and further research directions are outlined.

2. PROBLEM DESCRIPTION

The PDTW can be considered as a routing network, represented by a directed graph $G\{I,P,A\}$, connecting customer nodes $I = \{i_1, i_2, ..., i_n\}$ and depot nodes $P = \{p_1, p_2, ..., p_l\}$ through a set of directed edges $A = \{(i,j) \mid i,j \in (I \cup P)\}$. The edge $a_{ij} \in A$ is supposed to be the lowest cost route connecting node i to node j. At each customer location $i \in I$, a fixed load w_i is to be picked up or delivered by a single vehicle. A time window $[a_i, b_i]$ is defined for each customer i, where a_i is the earliest time and b_i is the latest time at which the service can start. Service time st_i at node i is also given.

A fleet of heterogeneous vehicles $V = \{v_1, v_2, ..., v_m\}$ with different capacities q_v located in a depot $p \in P$ is available to accomplish the required pick-up/delivery tasks. Each vehicle v must leave from the depot, perform corresponding pick-up and delivery orders without exceeding its capacity q_v at any time and then return to the same depot. In the present model, only one depot is considered, but the model can be easily extended to include multiple depots.

The route for vehicle v is a tour of nodes $r_v = (p, ..., i_j, i_{j+1}, ..., p)$ connected by directed edges belonging to A that starts and ends at depot p. Associated to the set of edges $a_{ij} \in A$, there is a pair of matrices $C = \{c_{ij}\}$ and $T = \{t_{ij}\}$ denoting the travel cost and the travel time from node i to node j, respectively. It is assumed that distances and times defined in the problem satisfy the triangular inequality, i.e. $c_{ik} + c_{kj} \geq c_{ij}$ and $t_{ik} + t_{kj} \geq t_{ij}$ $\forall i,j,k \in (I \cup P)$.

A set of work orders $O = \{(i,j) \mid i \neq j;\ i, j \in (I \cup P)\}$ is defined among customers ($O \subseteq (I \cup P) \times (I \cup P)$) to relate pick-up and delivery locations. Thus, if vehicle v is assigned to serve the order (i,j), denoted as o_{ij}, it must visit customer i to pick up the corresponding load w_i before visiting customer j, where a quantity w_j is to be delivered. In any case, both customers i and j related through an order o_{ij} should be assigned to the same vehicle's v route (5).

The proposed mathematical formulation, based on the MILP formulation presented in (Dondo and Cerdá 2007), requires defining two different sets of binary variables:

- The allocation variable Y_{iv} to assign vehicle $v \in V$ to customer $i \in I$, taking value 1. Otherwise, it is 0.
- The precedence variable S_{ij} to denote that customer $i \in I$ is visited before ($S_{ij} = 1$) or after ($S_{ij} = 0$) customer $j \in I$. It should be noted that this approach uses the notion of generalised predecessor rather than direct predecessor.

In the present model, PDTW has been divided into two subproblems, concerning resource's allocation and vehicle's routing separately. Allocation variable Y_{iv} is determined by solving the allocation subproblem, while precedence variable S_{ij} is set on the routing part. Furthermore, in the routing subproblem, values obtained for Y_{iv} variables are used. Thus, the allocation variable Y_{iv} is shared between both subproblems.

Allocation and routing subproblems have been formulated separately according to the formalisms to be used. Thus, Constraint Programming (CP) paradigm has been used to partially define the allocation subproblem, while routing calculation has been tackled from a MILP perspective.

The proposed allocation subproblem has been formulated as follow:

$$\min \sum_{v \in V} cf_v B_v \tag{1}$$

s.t.

$$occurrences(1, Y_{iv}, 1) \quad \forall i \in I \mid i > 1 \tag{2}$$

$$\sum_{i \in I} w_i Y_{iv} = 0 \quad \forall v \in V \tag{3}$$

$$w_i Y_{iv} \leq q_v \quad \forall i \in I, v \in V \tag{4}$$

$$Y_{iv} = Y_{jv} \quad \forall o_{ij} \in O, v \in V \tag{5}$$

where cf_v is the fixed cost associated to using vehicle v.

The objective function (1) aims to minimise the number of vehicles used in the solution. A boolean variable B_v has been defined for each vehicle $v \in V$ to determine whether v is used in the solution ($B_v = 1$) or not ($B_v = 0$), according to the expression

$$B_v n \geq \sum_{i \in I} Y_{iv} \tag{6}$$

being n the total number of customers.

Constraint (2) states that every customer node $i \in I$ must be serviced by a single vehicle $v \in V$. Equation (2) is a CP global constraint equivalent to the linear expression

$$\sum_{v \in V} Y_{iv} = 1 \quad \forall i \in I \mid i > 1 \tag{7}$$

Capacity constraint (3) states that the overall load along the route of a vehicle $v \in V$ should be zero. Thus, all vehicles should be empty when departing or finishing their respective routes.

Expression (4) ensures that vehicle v assigned to serve customer i has enough capacity to pick up the corresponding load w_i. This constraint is meaningless for delivery locations, since their demand is always non-positive.

The MILP formulation for the routing subproblem is presented next:

$$\min \sum_{v \in V} (CV_v + c_t TV_v + \rho_v \Delta T_v) + \sum_{i \in I} \rho_t (\Delta a_i + \Delta b_i) \tag{8}$$

s.t.

$$C_i \geq c_{pi} Y_{iv} \quad \forall i \in I, v \in V \tag{9}$$

$$C_j \geq C_i + c_{ij} - M_c(1 - S_{ij}) - M_c(2 - Y_{iv} - Y_{jv}) \tag{10a}$$

$$C_i \geq C_j + c_{ji} - M_c S_{ij} - M_c(2 - Y_{iv} - Y_{jv}) \tag{10b}$$

$$\forall v \in V; \forall i, j \in I \mid i < j$$

$$CV_v \geq C_i + c_{ip} - M_c(1 - Y_{iv}) \quad \forall i \in I, v \in V \tag{11}$$

$$T_i \geq t_{pi} Y_{iv} \quad \forall i \in I, v \in V \tag{12}$$

$$T_j \geq T_i + st_i + t_{ij} - M_t(1 - S_{ij}) - M_t(2 - Y_{iv} - Y_{jv}) \tag{13a}$$

$$T_i \geq T_j + st_j + t_{ji} - M_t S_{ij} - M_t(2 - Y_{iv} - Y_{jv}) \tag{13b}$$

$$\forall v \in V; \forall i, j \in I \mid i < j$$

$$TV_v \geq T_i + st_i + t_{ip} - M_t(1 - Y_{iv}) \quad \forall i \in I, v \in V \tag{14}$$

$$\Delta a_i \geq a_i - T_i \quad \forall i \in I \tag{15}$$

$$\Delta b_i \geq T_i - b_i \quad \forall i \in I \tag{16}$$

$$\Delta T_v \geq TV_v - tv_v^{\max} \quad \forall v \in V \tag{17}$$

$$Q_i \leq q_v + M_{pd}(1 - Y_{iv}) \quad \forall i \in I, v \in V \tag{18}$$

$$Q_i \geq w_i + \sum_{j \neq i; j \in I} w_i B_{ij} \quad \forall i \in I \tag{19}$$

where c_t is the cost per unit time; CV_v and TV_v are the total distance and time for vehicle v, respectively; C_i and T_i are the cost and travel time from the depot to node i, respectively; Q_i is the load of vehicle v at customer i; Δa_i and Δb_i are time window violations at customer i; ΔT_v is the time window violation for vehicle v; B_{ij} is a boolean variable; M_c, M_t and M_q are large positive numbers.

Problem's objective function (8) aims to minimise the overall service expenses, including travelling distance and time costs. Moreover, time window violations are included in the last two terms, penalizing non-fulfilment on either the maximum allowed working time (ΔT_v) or customers' time windows (Δa_i and Δb_i).

Equation (9) states that the cost of travelling from depot p to customer i (C_i) should be greater than or equal to the least travel cost from the depot to node i (c_{pi}). This constraint becomes binding iff customer i is the first visit included in vehicle's v route.

Constraint (10a) states that the distance based travel cost from the depot to customer j (C_j) should be greater than C_i by at least c_{ij} whenever customers i and j are included in the same route ($Y_{iv} = Y_{jv} = 1$, for some vehicle v) and i is visited first ($S_{ij} = 1$). In case customer j is visited earlier ($S_{ij} = 0$), the reverse statement (10b) holds. Constraints (10a) and (10b) both become redundant if nodes i and j are serviced by different vehicles. Furthermore, C_j values are determined only by previous visit's cost, due to triangular inequality ensures that C_i values are monotonically increasing. In addition, the latter prevents from cycles appearing in a feasible solution without adding specific constraints.

Expression (11) states that the overall travelling cost incurred by vehicle v (CV_v) must always be greater than, or equal to, the travelling expenses from the depot to any customer i (C_i) by at least the amount c_{ip}. Indeed, triangular inequality guarantees that the last node visited by vehicle v is the one finally binding the value of CV_v.

Constraints (12)-(14) are time equivalents to distance based cost constraints (9)-(11). Thus, properties described above still hold since expressions are only modified by adding the service time at customer i (st_i).

Time windows can be hard (HTW) or soft (STW). When the time windows are regarded as hard constraints, expression (15) states that a vehicle cannot start the service at the assigned customer i before the earliest time a_i by simply making $\Delta a_i = 0$. In turn, constraint (16) prohibits to start the service at node i after the allowed latest time b_i by setting $\Delta b_i = 0$. In the STW case, time window constraints may be violated at a finite cost (ρ_t) and the vehicle may start the service at node i before time a_i. In such a case, variables Δa_i and Δb_i stand for the size of time windows violations and are determined by expressions (15) and (16), respectively.

Constraint (17) applies in case the maximum allowed working time tv_v^{max} is regarded as a soft constraint that may be violated at some penalty cost (ρ_v). Otherwise, $\Delta T_v = 0$ and TV_v should not be greater than tv_v^{max}.

Expression (18) ensures that the current load at customer i never exceeds assigned vehicle's capacity. This constraint only becomes active in pick-up locations, since demands at drop-off nodes are non-positive. Therefore, if a pick-up location with an associated positive load fulfils expression (18), it is also accomplished by consequent delivery destinations.

Constraint (19) allows the current load to be traced at each visited customer. This expression states that it should be greater than, or equal to, the sum of all previous visits and the corresponding demand (w_i). A boolean variable (B_{ij}) is introduced to determine whether a customer is visited previously in the same route or not. Its value is given by the expression

$$B_{ij} \geq B_{ij}^v - S_{ij} \quad \forall i, j \in I, v \in V \tag{20}$$

where B_{ij}^v is determined according to

$$B_{ij}^v \geq Y_{iv} + Y_{jv} \text{-}1 \quad \forall i, j \in I, v \in V \tag{21}$$

3. HYBRID APPROACH

The model presented in the previous section has been programmed using the software ECL^{i}PS^e. It has been implemented using *eplex* and *ic* libraries, both included in the ECL^{i}PS^e package. The *eplex* library provides an interface to use an external mathematical programming (LP, MIP or quadratic) solver from within ECL^{i}PS^e. In this particular case, only default CPLEX solver (ILOG 2001) has been used. The *ic* library contains a general interval propagation solver which can be used to solve problems over both real and integer variables. Therefore, constraint propagation mechanisms are included in this library.

As a previous stage, complete CVRP, VRPTW and PDTW models (i.e. treating allocation and routing subproblems together) have been defined in *eplex* and *ic*, independently. However, separated implementations are not able to solve some problems or have proved to be quite inefficient. Thus, a hybrid approach has been adopted to tackle the PDTW problem, in order to increase efficiency but loosing optimality.

In the final implementation, different solvers have been used to tackle each subproblem. First, *ic* has been used to solve the allocation subproblem. Second, *eplex* solver is used in the routing subproblem. Both solvers share the allocation variable Y_{iv} and interact according to patterns described on (Apt and Wallace 2007). However, algorithm's structure allows swapping solvers with few changes. *Ic* and *eplex* could be easily interchanged, so *ic* would be used to solve the routing subproblem and *eplex* would provide an optimal solution for the allocation step. With this purpose, only expression (2) should be changed to its linear version (7). *Ic* functions permit to solve the proposed routing subproblem formulation, but lacking a good efficiency. Moreover, incorporating different algorithms or solvers to the model would also lead to few changes in models' general structure and problem's formulation.

Hybrid algorithm's structure proposed to solve PDTW problems is summarized next:

1. *ic* solves the allocation subproblem
2. *eplex* solves the routing subproblem using Y_{iv} values found by *ic*
3. Check time windows:
   (a) If they are not fulfilled:
      (i) Add new constraint to allocation subproblem
      (ii) Go to step 1
4. Return solution found

It can be observed that, even though the routing subproblem is defined using STW, hybrid algorithm's behaviour corresponds to a HTW model. Time windows constraints are checked after solving the routing subproblem. If they are not fulfilled for a particular route, a new constraint is added to the allocation subproblem, so all customers originally assigned to that route cannot be allocated together in the next iteration:

$$atmost(n_v - 2, Y_{rv}, 1) \tag{22}$$

This constraint states that at least a pick-up/delivery pair has to be removed from vehicle's v route (r_v). This restriction is performed by forcing Y_{iv} variables ($Y_{rv} = \{Y_{iv} \mid i \in r_v \subseteq I\}$) to take a maximum of n_v-2 times the value 1, being n_v the current number of customers assigned to route r_v. Expression (22) is a CP global constraint equivalent to the linear expression (23), used in case solvers are swapped.

$$Y_{av} + Y_{bv} + ... + Y_{lv} \leq \sum_{i \in r_v} Y_{iv} - 2 \quad \forall a, b, ..., l \in r_v \tag{23}$$

Allocation constraint (22) may be added according to two different criteria. First, it may be added whenever a vehicle has to wait before starting the corresponding service (arriving to a customer before the lower time window bound a_i) or finishing too late (after the upper bound b_i). On the other hand, constraint (22) may be only added if a vehicle violates any upper bound among customers included in its route. In this case, routes violating only lower bounds, i.e. vehicles are forced to wait but they can perform the service within defined time windows, are supposed to be able to accept new work orders to fill time gaps, so no additional constraints (22) are considered.

Once constraint (22) is added, *ic* solver is called again to rearrange customers. Then, *eplex* optimizes routes using new Y_{iv} values. The process finishes whether a solution fulfilling time windows constraints is found or *ic* solver fails to find a feasible solution for the allocation subproblem. Thus, this algorithm only succeeds if a solution for the HTW case exists. However, the algorithm may be easily modified to track results at each iteration, so it can return a previous solution if it fails.

It should be noticed that, every time a new constraint (22) is added to the allocation subproblem, a pair of pick-up/delivery locations is to be removed from the set of customers assigned to a route. However, it does not specify which particular work order has to be removed from the route. This fact, combined with the simplicity of the objective function (1), may lead to a lack of efficiency due to a random selection of customers to be removed. Some future work could be addressed to define a different objective function or heuristics for the allocation subproblem, in order to reduce this problem's impact.

3.1. Time windows pre-processing

Aiming to increase algorithm's efficiency, time windows defined for each customer are pre-processed to add additional constraints to the allocation subproblem. Depending on the problem instance, some time windows may be overlapped, so a vehicle cannot serve those customers without violating their respective working times. Time windows pre-processing is used to detect this situation and add constraints defining which customers are incompatible to be in the same route.

Two customers are determined to be incompatible if a vehicle cannot perform both services within their respective time windows. Considering a vehicle arriving to customer $i \in I$ and starting the service at a time corresponding to the lower bound of its time window (a_i), it cannot be visited first in the same route than a second customer $j \in I$ if the time cost to reach j from i exceeds former's time window upper bound (b_j), i.e. $a_i + st_i + c_{ij} > b_j$. If this expression also holds swapping indexes i and j, then customers i and j cannot lead to a feasible solution fulfilling time windows when they are included in the same route. Therefore, an additional constraint (24) is defined in the allocation subproblem, so i and j cannot be assigned to the same vehicle v (i.e. $Y_{iv} + Y_{jv} \leq 1 \quad \forall v \in V$):

$$atmost(1,[Y_{iv},Y_{jv}],1) \quad \forall v \in V \tag{24}$$

Constraint (24) is defined for each pair of incompatible customers. Thus, a first approach of the number of vehicles and routes to be calculated is obtained by classifying customers into groups where time windows constraints can be fulfilled. Moreover, time windows pre-processing helps to avoid the exploration of some unfeasible solutions, since it provides a more constrained allocation subproblem. Indeed, in the best case it may provide allocation subproblem's solution.

3.2. Routing subproblem decomposition

Routing subproblem is tackled using allocation variable Y_{iv} values obtained from the allocation subproblem. Although customers may be assigned to different vehicles, i.e. different routes, the optimization process used in the presented hybrid model performs the calculation over all routes at the same time. Thus, the number of variables that *eplex* solver should take into account may grow according to problem's instances. This fact may increase dramatically the computation time. Furthermore, time windows violations can only be detected at the end of the process.

In order to reduce the computation time, the proposed routing problem decomposition provides a method to calculate routes separately. With this purpose, an independent routing subproblem is solved for each vehicle, reducing problem's dimensions and improving algorithm's computation time. Moreover, time windows can be checked after each route is calculated, so a partial state that leads to an unfeasible global solution can be detected. Therefore, the routing subproblem decomposition may avoid the exploration of unpromising regions.

A set of customers is assigned to a vehicle v in the allocation process, defining Y_{iv} values. Then, distance and time submatrices can be obtained, including only those customers assigned to a particular vehicle and defining a smaller problem instance. On the other hand, routing subproblem's objective function (8) has been modified to take into account a single vehicle. Thus, the original routing subproblem is decomposed into several smaller instances according to the number of routes to be calculated, reducing the required computation time. Eventually, the total cost associated to the routing process corresponds to the sum of all vehicles' objective function values.

Furthermore, expressions (9)-(18) are simplified by removing all references to Y_{iv} values, since only assigned customers are considered (i.e. $Y_{iv} = 1$ always). Therefore, the number of constraints to be evaluated every time variable S_{ij} is labelled is clearly reduced, increasing algorithm’s efficiency. Indeed, constraint (19) can be modified so it only depends on precedence variable S_{ij} values:

$$Q_i \geq w_i + \sum_{j \in I} w_i (1 - S_{ij}) \quad \forall i \in I \tag{25}$$

Time windows can be checked after each route calculation or after all routes are determined. The former may save computation time, since all routes may not have to be calculated at each iteration. The latter may not provide such a computation time reduction, but a complete solution may be obtained at each iteration. In both cases, an additional constraint (22) is added to the allocation subproblem for every route whose assigned customers’ time windows are violated.

In the proposed model, time windows are checked after all routes have been calculated. Performance is improved due to smaller instances are tackled. At the same time, a complete solution is calculated at each iteration, even though time windows are not fulfilled, so a response may be provided if the allocation process fails after adding new constraints (22). Considering time windows checking after each route calculation would make this process more complex, since some routes would have to be recalculated after the allocation process fails.

4. APPLICATION AND RESULTS

The model presented in previous sections has been tested on small PDTW problem instances. These test cases have been defined aiming to detect errors and ensure all implementations provide expected results. Finally, a real-case based problem (Busquets et al. 2005) with a reasonable number of customers and work orders has been defined. Unfortunately, the model could not be tested on PDTW benchmark problems, since there is not a set of well accepted benchmark instances, although some efforts have been addressed in this direction (Lau and Liang 2001).

PDTW instances defined to test algorithm’s validity have the following characteristics:

- 6 customers grouped in 3 work orders, 1 depot and 2 vehicles with different capacities. Optimal solution’s cost is 483 and 2 vehicles are needed.
- 7 customers grouped in 5 work orders, where 4 of them share the same pick-up location, 1 depot and 2 vehicles with different capacities. Optimal solution’s cost is 266 and 1 vehicle is needed.
- 33 customers grouped in 16 work orders, where pick-up locations are shared, 1 depot and 3 vehicles with the same capacity. Optimal solution’s cost is 2825.9 and 3 vehicles are needed.

Table 1 show results obtained for different implementations and solvers combinations. In all cases, obtained solutions provide the exact number of vehicles required in the optimal case. Deviation from optimal solution is only indicated if the algorithm has not reached the optimum, otherwise it is 0%. Blanks correspond to those cases where the algorithm could not find a solution in a reasonable amount of time.

Table 1: Results Obtained for PDTW Instances

Solver	**6 cust.**	**7 cust. same pp**	**33 cust.**	
	Sol.	**Sol.**	**Sol.**	**Dev.**
eplex	483 (0.17s)	266 (1.50s)	-	-
ic	-	266 (3.02s)	-	-
eplex-ic	483 (0.23s)	266 (0.31s)	-	-
eplex-ic + TWP	483 (0.13s)	266 (0.34s)	-	-
eplex-ic + routes	483 (0.19s)	266 (0.47s)	-	-
eplex-ic + TWP + routes	483 (0.14s)	266 (0.44s)	-	-
ic-eplex + TWP	483 (0.16s)	**266 (0.25s)**	3134.1 (9.23s)	11%
ic-eplex + TWP + routes	**483 (0.08s)**	266 (0.33s)	**3134.1 (8.59s)**	11%

First two rows present results for single-solver initial models. It can be observed that *eplex* is more efficient than *ic*'s implementation, due to the model is more suitable for a MILP solver and does not correspond to a CP specific formulation. Next rows show results obtained for the proposed hybrid model combined with presented performance improvements, i.e. time windows pre-processing and routing subproblem decomposition. Several tests have been done interchanging solvers' tasks. First four results correspond to using *eplex* as allocation subproblem's solver and *ic* to tackle the routing task. Reverse assignation achieve results presented in last rows. It can be noticed that all models reach the optimum for small PDTW instances. On the other hand, 33-customers problem is only solved by *ic-eplex* hybrid algorithms, although the optimal solution is not reached. Computation times are also shown, with lowest values highlighted for each problem. In all cases, best calculation times are obtained for the *ic-eplex* model described in the present paper. It should be remarked that, in the 7-customers problem with shared pick-up locations, neither time windows pre-processing nor routing subproblem decomposition provide a significant computation time decrease. In this particular case, a single vehicle is needed to serve all customers, so only one route has to be configured. Thus, time may be spent checking corresponding time windows without adding additional constraints. Moreover, despite all models have to calculate a single route including same customers, routing subproblem decomposition forces the algorithm to dynamically define the *eplex* instance after solving allocation subproblem, i.e. additional time is to be spent on checking variables' values.

5. CONCLUSIONS AND FUTURE RESEARCH

The present paper introduces the developed study about the PDTW, a well known NP-Hard problem. It presents a complete model, decomposed into two subproblems based on CP and MILP paradigms, to make it suitable to adopt a hybrid approach.

Model's implementation has been described, as well as general interaction patterns between solvers used to tackle the problem: *ic* and *eplex*. Moreover, performance improving techniques have been introduced, such as time windows pre-processing and routing subproblem decomposition. This hybrid approach has proved to be more efficient than those based on a single solver. In addition, algorithm's structure and problem's definition permit to easily interchange or substitute these solvers. Thus, some future efforts could be focused on developing specific methods, such as Lagrangian Relaxation, to tackle each subproblem more efficiently.

This approach has demonstrated to be suitable to solve different problems in a reasonable computation time. However, the algorithm may not guarantee finding the optimal solution. Reasons might be found in subproblems decomposition, as well as constraint addition mechanisms. Further research is to be addressed in these directions to improve solutions' quality while keeping a low calculation time.

REFERENCES

Apt, K. and Wallace, M., 2007. *Constraint Logic Programming using ECL^{i}PS^e*. Cambridge University Press.

Bräysy, O. and Gendreau, M., 2005. Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science*, 39: 104-118.

Bräysy, O. and Gendreau, M., 2005. Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science*, 39: 119-139.

Busquets, S., Vilalta, L., Piera, M.A., Guasch, T. and Narciso, M., 2005. Specification of metaheuristics in colored petri nets models to tackle the Vehicle routing problem. *Proceedings of the International Mediterranean Modelling Multiconference (I3M'05)*. October 20-22, Marseille (France).

Dondo, R. and Cerdá, J., 2007. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176: 1478-1507.

ILOG, 2001. *ILOG CPLEX 7.1 User's Manual*. ILOG France.

Lau, H. and Liang, Z., 2001. Pickup and delivery with time windows: algorithms and test case generation. *Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*, pp.333-340. November 7-9, Dallas (Texas, USA).

Savelsbergh, M., 1985. Local search in routing problems with time windows. *Annals of Operations Research*, 4: 285-305.

Savelsbergh, M. and Sol, M., 1995. The general pickup and delivery problem. *Transportation Science*, 29: 17-29.

AUTHORS BIOGRAPHY

Daniel Guimarans is a PhD Student and Assistant Teacher at the Telecommunication and System Engineering Department at the Universitat Autònoma de Barcelona. He is an active member of the Logisim Research Group. His research interests are constraint programming, simulation and hybridisation of different techniques to solve industrial combinatorial problems.

Juan José Ramos is an Associated Professor at the Telecommunication and System Engineering Department at the Universitat Autònoma de Barcelona. He is coordinator and an active member of the Logisim Research Group. His research interests are modelling, simulation and optimisation of logistics problems.

Mark Wallace holds a chair in the Faculty of Information Technology at Monash University. He is director of the CTI-Monash Centre for Research in Optimisation and belongs to the Centre for Research in Intelligent Systems and the Optimisation and Constraint Solving Research Group. He is also involved in the NICTA Constraint Programming Platform project ("G12"). His research interests are focused on constraint programming, hybridisation of different techniques and its application to industrial combinatorial optimisation problems.

Daniel Riera received in 2006 his Computer Science PhD in the Universitat Autònoma de Barcelona. He is director of the Computer Science department in the Universitat Oberta de Catalunya and researcher of the GRES-UOC. Currently, he is co-director of the HAROSA (Hybrid Algorithms for solving Realistic rOuting, Scheduling and Availability problems) knowledge community. His research interests are constraint programming, simulation and techniques hybridisation to solve industrial combinatorial problems.