Chapter 4

BRANCH-AND-PRICE HEURISTICS: A CASE STUDY ON THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Emilie Danna
Claude Le Pape

Abstract Branch-and-price is a powerful framework to solve hard combinatorial problems. It is an interesting alternative to general purpose mixed integer programming as column generation usually produces at the root node tight lower bounds (when minimizing) that are further improved when branching. Branching also helps to generate integer solutions, however branch-and-price can be quite weak at producing good integer solutions rapidly because the solution of the relaxed master problem is rarely integer-valued. In this paper, we propose a general cooperation scheme between branch-and-price and local search to help branch-and-price finding good integer solutions earlier. This cooperation scheme extends to branch-and-price the use of heuristics in branch-and-bound and it also generalizes three previously known accelerations of branch-and-price. We show on the vehicle routing problem with time windows (Solomon benchmark) that it consistently improves the ability of branch-and-price to generate good integer solutions early while retaining the ability of branch-and-price to produce good lower bounds.

1. Introduction

Column generation is a powerful framework to solve hard optimization problems. It operates with a master problem that consists of a linear problem on the current set of columns, and a subproblem that iteratively generates improving columns. In case the master problem contains integrality constraints on some of its variables, column generation and branch-and-bound are combined: This is called branch-and-price, see Barnhart et al. (1998) for a general introduction. Branch-and-price pro-

vides the user both with a lower bound (when minimizing, as assumed throughout this paper) and integer solutions. Branch-and-price is known for providing tight lower bounds but it has sometimes difficulties to generate rapidly good solutions because the linear relaxation of the master problem rarely has an integer solution.

Local search (Aarts and Lenstra, 1997; Voß et al., 1999) is a completely different optimization technique with opposite properties. Local search algorithms use operators to define a neighborhood around a given solution or set of solutions. This subregion of the search space is then explored to iteratively generate better solutions and various strategies such as metaheuristics are used to move from one neighborhood to the next so as to escape local minima. Local search algorithms are notoriously effective at generating quickly excellent solutions. However, they do not provide the user with a lower bound on the objective. Hence the difference between the solution obtained and the optimal solution cannot be estimated and the user does not know if more time should be devoted to reach a better solution.

In this paper, we present a general cooperation scheme between branch-and-price and local search that improves the ability of branch-and-price to generate good integer solutions early while retaining the ability of branch-and-price to produce tight lower bounds.

In order to test this general hybrid scheme, we apply it to the vehicle routing problem with time windows (VRPTW). A number of industrial optimization problems are variations of the vehicle routing problem (VRP), which can be summarized as follows: Given a set of customers that each demand some amount of goods, a set of vehicles with given capacity that must start from and return to a depot, and known distances between all customers and the depot, and every pair of customers, the objective is to establish for each vehiclenn an ordered list of customers to visit so as to minimize the overall distance travelled and sometimes the number of vehicles needed. A classical additional constraint is to specify time windows that restrict the time of the day at which each customer can be served: This defines the vehicle routing problem with time windows. Cordeau et al. (2002) review different methods to solve it. Among exact methods, branch-and-price has recently been applied with success to this problem, see for example Desrochers et al. (1992); Kohl et al. (1999); Larsen (1999); Cook and Rich (1999); Kallehauge et al. (2001); Irnich (2001); Rousseau et al. (2002); Chabrier et al. (2002); Chabrier (2003); Irnich and Villeneuve (2003). Local search algorithms are also popular for solving the VRPTW, see for example Rochat and Taillard (1995); Homberger and Gehring (1999); Gambardella et al. (1999);

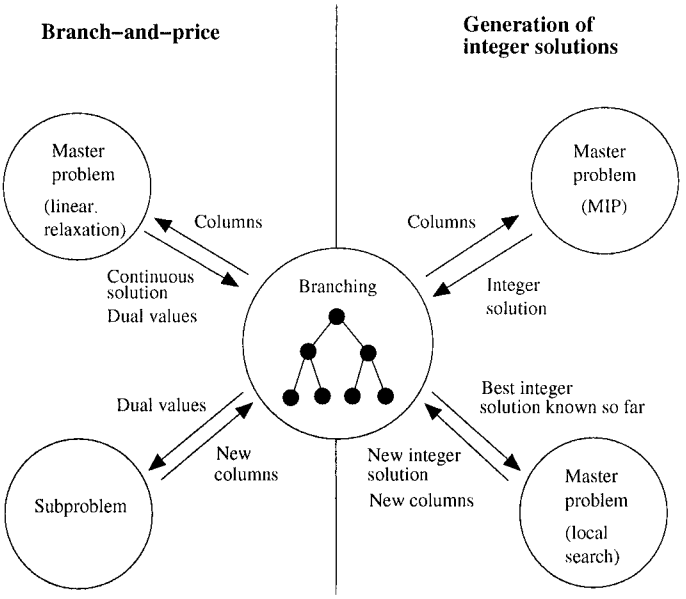

Figure 4.1. Cooperation scheme.

Cordeau et al. (2001); De Backer et al. (2000); Bräysy and Gendreau (2003a,b).

The remainder of the paper is organized as follows. Section 2 presents our general cooperation scheme between branch-and-price and local search. Section 3 details how our general scheme is applied to the vehicle routing problem with time windows. Section 4 gives computational results and discusses why our hybrid scheme works. Finally, Section 5 summarizes our conclusions.

2. General cooperation between branch-and-price and local search

2.1 Description of the algorithm and discussion

Figure 4.1 presents our cooperation scheme between column generation and local search. The left hand side of the figure shows the usual relaxed master problem and subproblem of branch-and-price. Note that the subproblem could be solved by any optimization technique. On the right hand side two components for obtaining integer solutions are specified. First, a mixed integer programming (MIP) solver is called regularly on the master problem with the current set of columns without relaxing

the integrality constraints. If the MIP solver is called at the root node of the branch-and-price tree, the best integer solution found so far is used as the first solution of the MIP. If the MIP solver is called at a node further down the branch-and-price tree, the best integer solution found so far might not be valid for the branching decisions taken at that node, hence it cannot always be used as a first solution. The effort spent on solving the MIP is controlled with a time or node limit. When this limit is reached, the exploration of the branch-and-price tree is resumed. Secondly, local search is also called regularly to solve the master problem, its initial solution being the best integer solution found so far. Unlike the MIP solver, local search is not restricted to combining existing columns: Local search may not only provide better combinations of existing columns, but it may also introduce new columns. Hence the columns generated are more diverse which is likely to accelerate pricing, for example because it has thus greater chances to overcome degeneracy.

The strength of our hybrid scheme is diversification by means of different algorithms for solving the same problem. Branch-and-price obviously benefits from local search that is more effective at finding feasible solutions. But in turn, local search benefits from branch-and-price that provides it with diverse initial solutions. Indeed, the main difficulty of local search algorithms is to escape local minima. To overcome this difficulty, the strategy of various metaheuristics is to attempt to control a series of moves that increase the value of the objective function in order to reach a different and more promising region of the solution space. There exist even simpler diversification schemes that restart the same algorithm from the same initial solution but with different random seed initialization (see for example Alt et al., 1996, for a theoretical study of this strategy) or build a new initial solution as different as possible from the current local optimum in order to explore a hopefully different region of the solution space. In all cases, diversification is achieved at the cost of increasing the objective function. On the contrary, in our cooperation scheme, the mathematical programming component is a non-deteriorating diversification scheme for local search. Indeed, the upper bound for the master problem and the MIP cutoff are always updated with the value of the best feasible solution found so far. Hence, when the MIP solver finds a new integer solution or when the solution of the relaxed master problem is integer, it is by construction an improvement on the last local optimum found by local search: Diversification is achieved and the objective function is improved at the same time. This strategy has nonetheless a computational cost: solving a MIP is more expensive than classical diversification schemes.

Branch-and-price is an exact method, so it will in the end find the optimal integer solution. Therefore our cooperation scheme is not so useful when exploring the branch-and-price tree to optimality. However, this complete exploration may find good integer solutions only late in the computation. Our cooperation scheme helps to find good integer solutions at an earlier stage, which has numerous advantages. First, the user can stop optimizing as soon as satisfied with the quality of the integer solution found and use the truncated exploration of the branch-and-price tree as a powerful heuristic that also provides tight lower bounds. Next, good upper bounds are helpful to solve the subproblem more effectively, for example by allowing to eliminate arcs from the shortest path subproblem in the VRPTW case, see for example Hadjar et al. (2001); Irnich and Villeneuve (2003). A good upper bound may also reduce the number of iterations between master problem and subproblem at each node: Computing the so-called Lagrangean lower bound (LLB) while solving a tree node might allow to terminate the column generation process at that node before optimality, i.e. as soon as LLB is greater than the upper bound known so far, see for example Desrosiers and Lübbecke (2004). Finally, knowing a good upper bound early might help to explore only a relatively small number of nodes in the branch-and-price tree. Given a fixed branching strategy, a best-first exploration strategy guarantees that only the children nodes of a node with a lower bound smaller than the optimal objective value have to explored. In this sense, best-first search guarantees that a minimum number of nodes are explored. Knowing a good upper bound does not allow us to improve on this number. However, best-first search can fail to produce good integer solutions until the very end of the tree exploration, this is why other exploration strategies such as depth-first search are often preferred, although they lead to a higher number of explored nodes. Our cooperation scheme allows the user to choose a tree exploration strategy such as best-first search that explores a small number of nodes because our scheme doesn't rely only on branching to generate integer solutions.

2.2 Related work

The first algorithms related to the cooperation scheme just described are the so-called mixed integer programming heuristics, such as pivot-and-complement introduced by Balas and Martin (1980) or the diving heuristics described in Bixby et al. (2000). These heuristics are used in branch-and-bound (and branch-and-cut) to generate good integer solutions by taking heuristic decisions outside of the exploration of the tree when branching has difficulties in finding integer solutions. Our cooper-

ation scheme between branch-and-price and local search can be seen as a generalization to branch-and-price of this use of heuristics in branch-and-bound in so far as it achieves the same goal (generating integer solutions early without interfering with the tree exploration strategy) and new columns are also introduced while generating integer solutions.

Our cooperation scheme also relates to the three following accelerating strategies for branch-and-price reviewed in Desaulniers et al. (2002): Using local search to generate initial primal and dual solutions, generating further integer solutions for the master problem by rounding to 1 or to the next integer the fractional variables of its continuous relaxation, and post-optimizing with local search the best known integer solution after a given time limit. Our cooperation scheme is a generalization of these accelerating strategies for the two following reasons. First, in our cooperation scheme, local search is called throughout the branch-and-price search and not only at the beginning or at the end of the optimization process. As explained in the previous section, this allows for fruitful interactions between the two components. Secondly, any local search method can be used in our cooperation scheme—Not just simple rounding techniques. Very effective domain-specific heuristics can be used, as we will show on the vehicle routing problem with time windows.

Finally, it should be mentioned that another existing strategy for combining local search and branch-and-price is to solve the subproblem with a local search algorithm. In our cooperation scheme, the local search algorithm generates new columns when solving the master problem. New columns can also be generated by directly solving the subproblem with a local search algorithm, as described for example in Savelsbergh and Sol (1998); Xu et al. (2003).

2.3 Parameters settings

A fair amount of tuning can be required so as to know when and for how long MIP and local search should be called. It obviously depends on the problem, but here are a few basic rules. Solving completely the MIP formulation of the master problem is time consuming so a time or node limit should be set and the MIP solver should preferably be called when we guess it has a good chance to find an improved integer solution, for example when the integrality gap between the best known integer solution and the value of the current continuous relaxation is high, or when the number of integer-infeasible variables in the relaxed master problem is small. Local search should be called for post-optimization at least each time a new integer solution is found by MIP or when the continuous relaxation of the master problem is integer. If local search turns out

to find significantly more solutions than mathematical programming, it may be called more often, using as first solutions not only the solutions found by mathematical programming but also some of the solutions previously found by local search itself. A simple adaptive scheme can be used, decreasing or increasing the frequency and the computation time allotted to MIP and local search according to their respective success rates.

3. Application to the vehicle routing problem with time windows

We now present the specific branch-and-price model and the heuristics we used for applying the general cooperation scheme just described to the vehicle routing problem with time windows.

3.1 Branch-and-price model and solution techniques

We used the following common model (Cordeau et al., 2002) where each column corresponds to a feasible route. Let $\{1, \ldots, n\}$ be the set of customers. For each feasible route r, let x_r be the variable defined by:

$$x_r = \begin{cases} 1 & \text{if route } r \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

and let c_r be the cost of route r. The VRPTW is then written as:

$$\min \sum_{r \in R} c_r x_r \quad \text{s.t.} \tag{4.1}$$

$$\sum_{r \in R} \delta_{ir} x_r = 1, \quad \forall i = 1, \ldots, n \tag{4.2}$$

$$x_r \in \{0, 1\}, \quad \forall r \in R \tag{4.3}$$

where R is the set of all feasible routes with respect to the capacity and time windows constraints, and $\delta_{ir} = 1$ if customer i is visited by route r, and 0 otherwise.

3.1.1 Decomposition into master problem and subproblem.

The first difficulty of this model is that the number of feasible routes grows exponentially with the number of customers. We hence use column generation to generate columns on the fly. The model is decomposed into a master problem and in a subproblem. The master problem is

formulated as:

$$\min \sum_{r \in \widehat{R}} c_r x_r \quad \text{s.t.} \tag{4.4}$$

$$\sum_{r \in \widehat{R}} \delta_{ir} x_r = 1, \quad \forall i = 1, \ldots, n \tag{4.5}$$

$$x_r \in \{0, 1\}, \quad \forall r \in \widehat{R} \tag{4.6}$$

where $\widehat{R}$ is the set of already generated columns. We solve in fact the continuous relaxation of the master problem, replacing (4.6) with the constraints

$$0 \leq x_r \leq 1, \quad \forall r \in \widehat{R}. \tag{4.7}$$

The subproblem is the following:

$$\min_{r \in R} c_r - \sum_{i=1}^{n} \pi_i \delta_{ir}$$

where $(\pi_i)_{i=1}^n$ is the dual price associated with (4.5). The subproblem is to be interpreted as a constrained shortest path problem on the original graph, where each arc (i, j) is valued by its original cost (distance) minus the dual value π_i associated with its starting extremity i.

3.1.2 Branch-and-price and branching strategy. The second difficulty of this model is that, as stated in (4.6), x_r variables must take integer values in feasible solutions. Therefore the problem is solved with branch-and-price:

1 Start with an initial pool of columns, for example generated by a simple heuristic.

2 Solve the continuous relaxation of the master problem, replacing (4.6) with (4.7).

3 Solve the subproblem with the dual values updated at step 2, and attempt to generate several constrained shortest paths with negative reduced costs.

4 Iterate steps 2 and 3 until no more new routes with negative reduced cost can be generated.

5 If the solution of the continuous relaxation of the master problem is not integer, branch and iterate steps 2 and 3 at each node.

We use the following branching rule on arcs. Let x^* be the optimal solution of the relaxed master problem after the last subproblem iteration at the current node. If x^* is integer, no branching is necessary. If x^* is not integer, let $t_0 = \{i_0 = \text{depot}, i_1, i_2, \ldots, i_{p+1} = \text{depot}\}$ be the route such that x_{t_0} is the variable with most fractional value in x^*. $x_{t_0} < 1$, hence for each $k \in \{1, \ldots, p\}$, there exist other routes that cover i_k and take a non-zero value in x^*. For each route t such that $x_t^* > 0$ and that shares at least one node with t_0, there exists $q \in \{1, \ldots, p\}$ such that t covers i_q but does not take arc (i_q, i_{q+1}) or arc (i_{q-1}, i_q). Indeed, every route in $\widehat{T}$ is unique, hence t and t_0 can have a common subsequence of nodes but necessarily differ from each other by at least one arc (which initial or final extremity may be the depot). So, we enumerate the columns already generated and choose the first route t such that $x_t > 0$ and that shares at least one node with t_0. Then we choose (i_q, i_{q+1}) as branching arc where $q \in \{1, \ldots, p\}$ is the smallest index such that $(i_q, i_{q+1}) \notin t$ (or we choose arc $(i_0 = \text{depot}, i_1)$ if $(i_q, i_{q+1}) \in t \ \forall q \in \{1, \ldots, p\}$). The child nodes are then created as follows. In one branch, arc (i_q, i_{q+1}) is forbidden. In the other branch, i_q and i_{q+1} are allowed to be taken in a route only if they are linked by arc (i_q, i_{q+1}). In other words, in the second branch, every arc (i_q, r) with $r \neq i_{q+1}$ and every arc (s, i_{q+1}) with $s \neq i_q$ are forbidden. This branching rule is very practical because it is easy to incorporate in the master problem and in the subproblem.

3.1.3 Solving the subproblem. The subproblem is solved with dynamic programming, with an adaptation of the label-based algorithm described in Desrochers (1988) so as to solve the *elementary* constrained shortest path problem. Details are given in a previous joint work with Alain Chabrier, see Chabrier et al. (2002); Chabrier (2003). The same idea was developed independently in Feillet et al. (2004). The motivation for generating only *elementary* constrained shortest paths in the subproblem is the following. If the distance used to compute the cost of routes conforms to the triangular inequality, then the optimal solution contains only elementary routes, whether cycles are allowed in the subproblem or not. Solving the non-elementary constrained shortest path is easier, so most column generation models in the literature allow cycles to be generated in the subproblem and sometimes add some mechanisms to partially eliminate non-elementary routes or improve the lower bound, see for example Houck et al. (1980); Kohl et al. (1999); Cook and Rich (1999); Irnich (2001); Irnich and Villeneuve (2003). These mechanisms are instrumental in solving instances with large time windows or with a large horizon because these instances are only loosely constrained by the

numerical data: It is entirely possible to build non-elementary routes and even to traverse cycles several times in the same route.

In short, the label-based algorithm used to solve the subproblem develops as follows. Partial paths starting from the depot and visiting a number of customers are built. As in a typical dynamic programming algorithm, dominated partial paths are gradually eliminated: If partial path p_1 and partial path p_2 both end at the same customer i, but p_1 arrives sooner at i, has smaller accumulated demand, and is less expensive than p_2, then partial path p_2 can be eliminated. Indeed, for every extension of p_2 to a complete path, p_1 could be extended in the same way and its extension would be less expensive than the extension of p_2. However, this dominance rule is no longer valid if we want to compute only *elementary* paths. Indeed, if the aforementioned extension of p_2 visits some customers that were already visited before i in p_1, then p_1 cannot be extended in the same way as p_2 because it would lead p_1 to visit these customers twice. We therefore change the dominance rule into the following: If partial path p_1 and partial path p_2 both end at the same customer i, but p_1 arrives sooner, has smaller accumulated demand, is less expensive than p_2, and if the set of customers visited by p_1 is a subset of the customers visited by p_2, then partial path p_2 can be eliminated. Refinements of this dominance rule are described in more details in Chabrier et al. (2002); Chabrier (2003). This implementation of elementary shortest path allowed us to solve to optimality 17 instances that were previously open (Chabrier et al., 2002; Chabrier, 2003), 9 of which have now been solved also by Irnich (2001); Irnich and Villeneuve (2003).

3.1.4 Acceleration strategies. Various well-studied accelerations of the above branch-and-price model allowed us to improve computational times. In particular, (4.5) is replaced by a set covering inequality:

$$\sum_{r \in \widehat{R}} \delta_{ir} x_r \geq 1, \quad \forall i = 1, \ldots, n. \tag{4.8}$$

As a consequence, integer solutions may cover some customers more than once, especially at the beginning of the branch-and-price process. Therefore, each time the MIP solver finds a new integer solution or the relaxed master problem produces an integer solution, we use a greedy heuristic that iteratively removes each customer visited more than once from all routes except from the route from which its removal would yield the smallest cost saving. This allows us to improve the integer solution and the resulting columns are also added to the column pool.

3.2 Heuristics

3.2.1 Building an initial solution. Heuristics are used for two different purposes. A first heuristic is used to build an initial solution. This initial solution can be as simple as the trivial solution "one customer per route". In our cooperation scheme and in the pure local search scheme, we start with the solution generated with the *savings* heuristic (Clarke and Wright, 1964; Paessens, 1988) adapted to the problem with time windows.

Heuristics are secondly used to improve on a given solution. We used two local search algorithms, a relatively simple one and a more sophisticated one. Our computational results will demonstrate the effectiveness of our cooperation scheme with these two different examples of local search, which leads us to think that our cooperation scheme is likely to be applied successfully in different settings. This will also allow us to show that even a simple local search algorithm can improve the ability of branch-and-price to generate good integer solutions early.

3.2.2 Large neighborhood search. We first implemented a Large Neighborhood Search (LNS) scheme based on constraint programming as described in Shaw (1998). Large Neighborhood Search proceeds by iteratively fixing some variables of the problem to their value in the current solution and solving a smaller subproblem on the rest of the variables. For the VRPTW, this amounts to removing a set of customers from the current solution and inserting them back again to build a better solution. First a small number of customers are released. If no better solution is found during a given number of iterations, the subproblem is enlarged, that is more customers are released simultaneously, see Shaw (1998) for details.

LNS turned out to be too slow for neighborhoods consisting of more than 20 customers. Therefore, in our pure LNS algorithm, we also use a restart mechanism for further diversification. When the size of the LNS neighborhood reaches 20, a quite different and possibly worse solution is built by an insertion heuristic. The customers are inserted in the "orthogonal" order of the current best solution: The first customer of each route is inserted, then the second customer of each route, etc. Each customer is inserted in its least expensive insertion point and additional routes are opened as needed. In the end, some customers are randomly moved from one route to another. The obtained solution is used as the next starting point for a new complete run of LNS.

3.2.3 Guided tabu search. In the second phase of our work, we decided to use a highly effective implementation of local search for

vehicle routing problems: ILOG DISPATCHER. The neighborhood used in this case is the union of all possible 2-OPT (Croes, 1958; Lin, 1965), Or-OPT (Or, 1976), Relocate (insert a customer in another route), Exchange (swap two customers of two different routes), and Cross moves (exchange the ends of two routes). The exact implementation is detailed in De Backer et al. (2000) and in Ilog Dispatcher User's Manual (2002). As for metaheuristics, we use in this case guided tabu search which is a mix of guided local search (GLS) and tabu search. Its implementation is described in De Backer et al. (2000); Ilog Dispatcher User's Manual (2002). GLS introduced by Voudouris (1997) is a metaheuristic that helps hill-climbing algorithms to escape local optima. It relies on optimizing an adaptively modified cost function based on the original cost function, but penalizing features that appear often in a solution. At each iteration, the penalized objective is first optimized using the hill-climbing algorithm. The penalized objective is then modified, increasing or decreasing the penalty of features according to their cost and to the number of iterations during which they have been penalized. This long-term memory mechanism enables to diversify the search.

However, in the cooperation scheme, the solutions found by the MIP solver or when the solution of the relaxed master problem is integer already ensure the long-term diversification of the local search component. A mechanism for short-term diversification is nonetheless needed and this is ensured by tabu search. Tabu search (Glover and Laguna, 1997) is a well-known effective metaheuristic. Basically, it escapes local optima by forbidding during a certain number of iterations (the tabu *tenure*) properties of moves recently performed or solutions recently visited, unless a certain *aspiration criterion* is validated—For example, the moves lead to a better solution than the best known solution so far. If the tabu tenure corresponds to a small number of iterations, this is a short-term memory mechanism.

De Backer et al. (2000) show that, as implemented in ILOG DISPATCHER for vehicle routing problems, Guided Tabu Search performs better than either simple Guided Local Search or simple Tabu Search.

4. Computational results

4.1 Benchmark

All computational testing described in the next sections have been performed on the well-known Solomon VRPTW instances introduced in Solomon (1987), on which exact and heuristic methods for solving the VRPTW are often tested. We adopted the conventions used by most exact methods: The objective is to minimize overall distance independently

of the number of vehicles used; distances and traveling times are determined by the Euclidean distance rounded downward at the first decimal place. The Solomon benchmark comprises two series of instances:

- series 1, the vehicle capacity is limited and the planning horizon is rather short;
- series 2 has vehicles with larger capacity and a longer planning horizon, which allows more customers to be served by the same route.

Hence, instances of series 1 are easier to solve because they are less combinatorial: The total number of feasible routes for these instances is smaller than for instances in series 2. Literature has so far concentrated on series 1, with some notable exceptions: Larsen (1999); Cook and Rich (1999); Kallehauge et al. (2001); Irnich (2001); Chabrier et al. (2002); Chabrier (2003); Irnich and Villeneuve (2003)—See Cordeau et al. (2002) for a general survey. Solomon instances are further divided in three groups: For "R" instances, customers are geographically randomly distributed; for "C" instances, customers are geographically clustered; and for "RC" instances, customers are alternatively random and clustered. Each instance is a 100-customer problem, from which a smaller problem is constructed taking into account only the first 50 customers.

Tables 4.1 and 4.2 give the solutions with which we will compare our results in Sections 4.3 and 4.5. These reference solutions are the best results known to us, taken from the literature or our own experiments (see next section for a precise description of our methods), as indicated in the *Origin* column. Solutions marked with * have been proved optimal. When the optimum is not known, we take the best known lower bound for series 1 (indicated in *italics*). But as no good lower bound is known for the open instances of series 2, we take the best known upper bound instead. The number of vehicles corresponding to each upper bound is given in parentheses.

4.2 Methods

Recall that the approaches compared are:

1 Our cooperation scheme between branch-and-price and local search (BP+LNS, BP+DISPATCHER). The MIP solver is called every 4 minutes with a 1-minute time limit. The local search algorithm is called for 10 seconds every 2 minutes and after a new and better integer solution has been found by branch-and-price. If the local search algorithm finds a new solution during a run, it is called immediately thereafter, again for 10 seconds. This very simple

Table 4.1. Reference solutions for series 1

Instance	50 customers		100 customers	
	Cost	Origin	Cost	Origin
C101	362.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
C102	361.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
C103	361.4 (5)*	Kohl et al. (1999)	826.3 (10)*	Kohl et al. (1999)
C104	358.0 (5)*	Kohl et al. (1999)	822.9 (10)*	Kohl et al. (1999)
C105	362.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
C106	362.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
C107	362.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
C108	362.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
C109	362.4 (5)*	Kohl et al. (1999)	827.3 (10)*	Kohl et al. (1999)
R101	1044.0 (12)*	Kohl et al. (1999)	1637.7 (20)*	Kohl et al. (1999)
R102	909.0 (11)*	Kohl et al. (1999)	1466.6 (18)*	Kohl et al. (1999)
R103	772.9 (9)*	Kohl et al. (1999)	1208.7 (14)*	Kohl et al. (1999)
R104	625.4 (6)*	Kohl et al. (1999)	962.3 (11)*	Irnich and Villeneuve (2003)
R105	899.3 (9)*	Kohl et al. (1999)	1355.3 (15)*	Kohl et al. (1999)
R106	793.0 (5)*	Kohl et al. (1999)	1234.6 (13)*	Cook and Rich (1999),Kallehauge et al. (2001)

Table 4.1 (continued).

Instance	50 customers		100 customers	
	Cost	Origin	Cost	Origin
R107	711.1 (7)*	Kohl et al. (1999)	1064.6 (11)*	Cook and Rich (1999), Kallehauge et al. (2001)
R108	617.7 (6)*	Kohl et al. (1999)	*919.9 (-)*	*Irnich and Villeneuve (2003)*
R109	786.8 (8)*	Kohl et al. (1999)	1146.9 (13)*	Cook and Rich (1999), Kallehauge et al. (2001)
R110	697.0 (7)*	Kohl et al. (1999)	1068 (12)*	Cook and Rich (1999), Kallehauge et al. (2001)
R111	707.2 (7)*	Cook and Rich (1999),Kallehauge et al. (2001)	1048.7 (12)*	Cook and Rich (1999),Kallehauge et al. (2001)
R112	630.2 (6)*	Cook and Rich (1999), Kallehauge et al. (2001)	*935.1 (-)*	*Cook and Rich (1999)*
RC101	944.0 (8)*	Kohl et al. (1999)	1619.8 (15)*	Kohl et al. (1999)
RC102	822.5 (7)*	Kohl et al. (1999)	1457.4 (14)*	Cook and Rich (1999), Kallehauge et al. (2001)
RC103	710.9 (6)*	Kohl et al. (1999)	1258 (11)*	Cook and Rich (1999), Kallehauge et al. (2001)
RC104	545.8 (5)*	Kohl et al. (1999)	1132.3 (10)*	Irnich and Villeneuve (2003)
RC105	855.3 (8)*	Kohl et al. (1999)	1513.7 (15)*	Kohl et al. (1999)
RC106	723.2 (6)*	Kohl et al. (1999)	*1356.1 (-)*	*Cook and Rich (1999)*
RC107	642.7 (6)*	Kohl et al. (1999)	1207.8 (12)*	Irnich and Villeneuve (2003)
RC108	598.1 (6)*	Kohl et al. (1999)	1114.2 (11)*	Irnich and Villeneuve (2003)

Table 4.2. Reference solutions for series 2

Instance	50 customers		100 customers	
	Cost	Origin	Cost	Origin
C201	360.2 (3)*	Cook and Rich (1999), Larsen (1999)	589.1 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)
C202	360.2 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)	589.1 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)
C203	359.8 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)	588.7 (3)*	Kallehauge et al. (2001)
C204	350.1 (2)*	Kallehauge et al. (2001)	588.1 (3)*	Irnich and Villeneuve (2003)
C205	359.8 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)	586.4 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)
C206	359.8 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)	586.0 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)
C207	359.6 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)	585.8 (3)*	Cook and Rich (1999), Kallehauge et al. (2001)
C208	350.5 (2)*	Cook and Rich (1999), Kallehauge et al. (2001)	585.8 (3)*	Kallehauge et al. (2001)
R201	791.9 (6)*	Cook and Rich (1999), Kallehauge et al. (2001)	1143.2 (8)*	Kallehauge et al. (2001)
R202	698.5 (5)*	Cook and Rich (1999), Kallehauge et al. (2001)	1029.6 (8)	BP+DISPATCHER
R203	605.3 (5)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	871.4 (6)	ILOG DISPATCHER
R204	506.4 (2)*	Irnich and Villeneuve (2003)	733.0 (5)	Røpke (2003)
R205	690.1 (5)*	Larsen (1999), Kallehauge et al. (2001)	951.9 (5)	Røpke (2003)
R206	632.4 (4)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	880.6 (4)	Røpke (2003)
R207	575.5 (3)	ILOG DISPATCHER, Røpke (2003)	794.0 (4)	Røpke (2003)

Table 4.2 (continued).

Instance	50 customers		100 customers	
	Cost	Origin	Cost	Origin
R208	487.7 (2)	ILOG DISPATCHER, Røpke (2003)	701.2 (3)	Røpke (2003)
R209	600.6 (4)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	855.7 (5)	Røpke (2003)
R210	645.6 (4)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	900.8 (6)	Røpke (2003)
R211	535.5 (3)*	Irnich and Villeneuve (2003), BP, BP+DISPATCHER	751.7 (3)	Røpke (2003)
RC201	684.8 (5)*	Larsen (1999), Kallehauge et al. (2001)	1261.8 (9)*	Kallehauge et al. (2001)
RC202	613.6 (5)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	1092.3 (8)*	Chabrier et al. (2002) Chabrier (2003)
RC203	555.3 (4)*	Chabrier et al. (2002), Chabrier (2003)	923.7 (5)	Røpke (2003)
RC204	444.2 (3)*	BP+DISPATCHER	783.5 (4)	Røpke (2003)
RC205	630.2 (5)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	1154.0 (7)*	Chabrier et al. (2002), Chabrier (2003)
RC206	610.0 (5)*	Irnich (2001), Chabrier et al. (2002), Chabrier (2003)	1051.1 (6)	Chabrier et al. (2002)
RC207	558.6 (4)*	Chabrier et al. (2002), Chabrier (2003)	966.3 (5)	Chabrier et al. (2002)
RC208	476.7 (3)	Chabrier et al. (2002)	777.3 (3)	Røpke (2003)

adaptive scheme allows us to call the local search algorithm more often if it succeeds, without slowing down too much the completion of the optimality proof after the optimal solution has been reached.

2 Almost the same branch-and-price method as in the hybrid scheme, but used without local search. The minor differences with the branch-and-price and MIP scheme used in the cooperation are twofold:

- The MIP solver is called more often (every 3 minutes instead of every 4 minutes) with the same 1-minute time limit for each run, so as to compensate for the lack of other heuristics to generate integer solutions.
- In "BP 1", the initial pool of columns is the trivial solution built with one customer per route. In "BP 2", branch-and-price starts from the solution generated with the *savings* heuristic, as in the hybrid scheme.

3 The same local search method as in the hybrid scheme (LNS or ILOG DISPATCHER), but used alone.

The parameters were chosen experimentally. We found out that it was more effective to call the MIP solver often and with a small time limit than less often with a longer time limit because failures of the MIP solver to produce new integer solutions appeared to come from the inexistence of improving columns rather than from the inability of the solver to optimize successfully the MIP model. Note that the allocation of time limits and call frequencies to the different components of the hybrid scheme renders the execution of the overall algorithm non-deterministic and computer-dependent. Yet such an allocation does make sense when, as is often the case in practice, the main objective is to obtain the best possible result in limited CPU time.

All results were obtained with a one hour time limit for each instance, on a Pentium IV-1.5 GHz with 256 Mb of RAM, using ILOG CPLEX 8.1.0, ILOG SOLVER 5.3 and ILOG DISPATCHER 3.3.

4.3 Quality of integer solutions

We now present the main results for the methods we have just described. Table 4.3 shows the quality of solutions obtained by each algorithm on each series of the Solomon benchmark, for 50-customer and 100-customer instances. The quality of solutions obtained by each algorithm is measured as the mean relative deviation (in %) between the reference solution of Tables 4.1 and 4.2, and the upper bound obtained

Table 4.3. Quality of solutions obtained.

Algorithm	Number of customers	Mean relative deviation (%)							Number of times optimality is reached (and proved)
		C1	R1	RC1	C2	R2	RC2	All	
BP 1	50	0.00	0.15	1.15	0.74	3.73	2.05	1.33	40 (37)
	100	0.00	2.40	6.17	6.50	8.63	5.26	4.77	17 (15)
BP 2	50	0.07	0.25	1.32	0.68	3.12	2.50	1.32	41 (37)
	100	0.29	2.37	6.05	7.80	7.76	5.60	4.86	17 (15)
Pure LNS	50	0.00	0.11	0.20	0.00	1.69	1.25	0.56	41 (-)
	100	0.00	2.96	3.93	2.61	5.85	7.18	3.74	12 (-)
Pure DISPATCHER	50	0.00	0.03	0.12	0.40	0.89	1.13	0.42	33 (-)
	100	0.09	1.06	2.36	0.00	0.62	1.66	0.94	13 (-)
BP+LNS	50	0.00	0.00	0.25	0.12	3.27	0.38	0.75	47 (34)
	100	0.00	1.70	3.62	3.04	6.08	3.93	3.07	19 (15)
BP+DISPATCHER	50	0.00	0.12	0.51	0.75	1.15	0.36	0.48	45 (38)
	100	0.00	1.47	4.46	0.26	4.13	2.56	2.17	20 (16)

by this algorithm. Recall that on series 2 the reference solutions are possibly sub-optimal upper bounds, hence the numbers given for this series are not necessarily upper bounds for the distance to the optimal solution. Table 4.3 also gives the number of instances for which each algorithm reaches and also proves (in parentheses) optimality. Recall that only pure branch-and-price and our cooperation scheme are able to produce optimality proofs. Note for comparison purposes that there are 56 instances in each 50-customer and 100-customer category. The optimal solution is known for 53 instances of the 50-customer category, and 38 instances of the 100-customer category. Figures 4.2 through 4.4 show the evolution of solution quality over time for all 100-customer instances.

Our first conclusions from this experimental data are the following. Combining local search and branch-and-price is consistently more effective than branch-and-price alone at obtaining good feasible solutions. On all series, on 100-customers and 50-customer instances, both our hybrids combining branch-and-price and local search (LNS or ILOG DISPATCHER) are as effective as or, in most cases, more effective than branch-and-price alone. The improvement obtained by combining branch-and-price and local search is most often correlated with the performance of the local search algorithm used alone. As expected, our simple LNS algorithm performs worse than the more sophisticated guided tabu search from ILOG DISPATCHER. In the same way, the cooperation scheme combining branch-and-price and LNS is outperformed in most series by the cooperation scheme combining branch-and-price and ILOG DISPATCHER. However, even a simple local search algorithm such as LNS can improve significantly the performance of branch-and-price.

On the contrary, for pure branch-and-price algorithms, starting from a pool of columns built by a simple heuristic ("BP 2") does not consistently

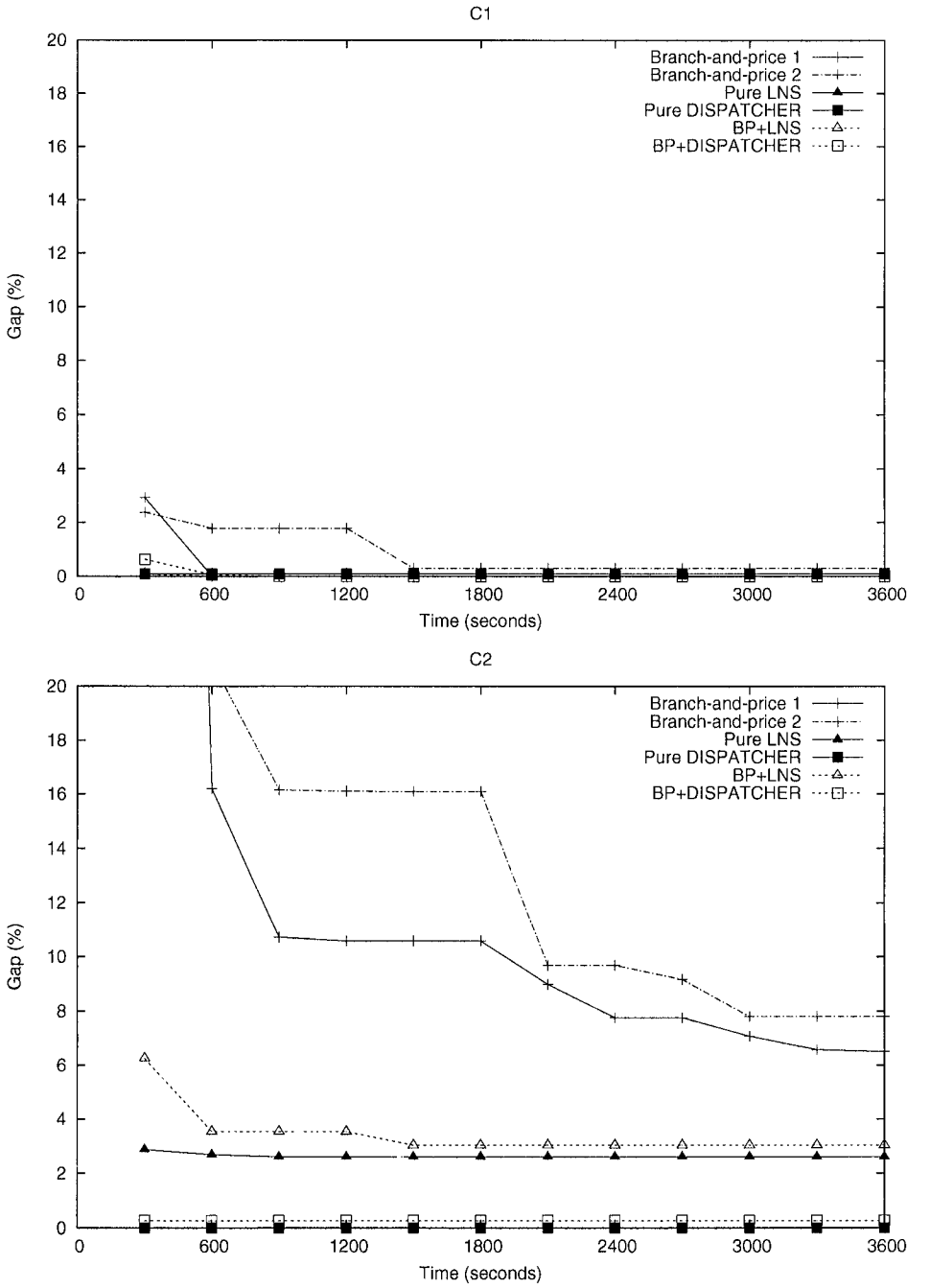

Figure 4.2. Evolution of solution quality over time for C series (100-customer instances).

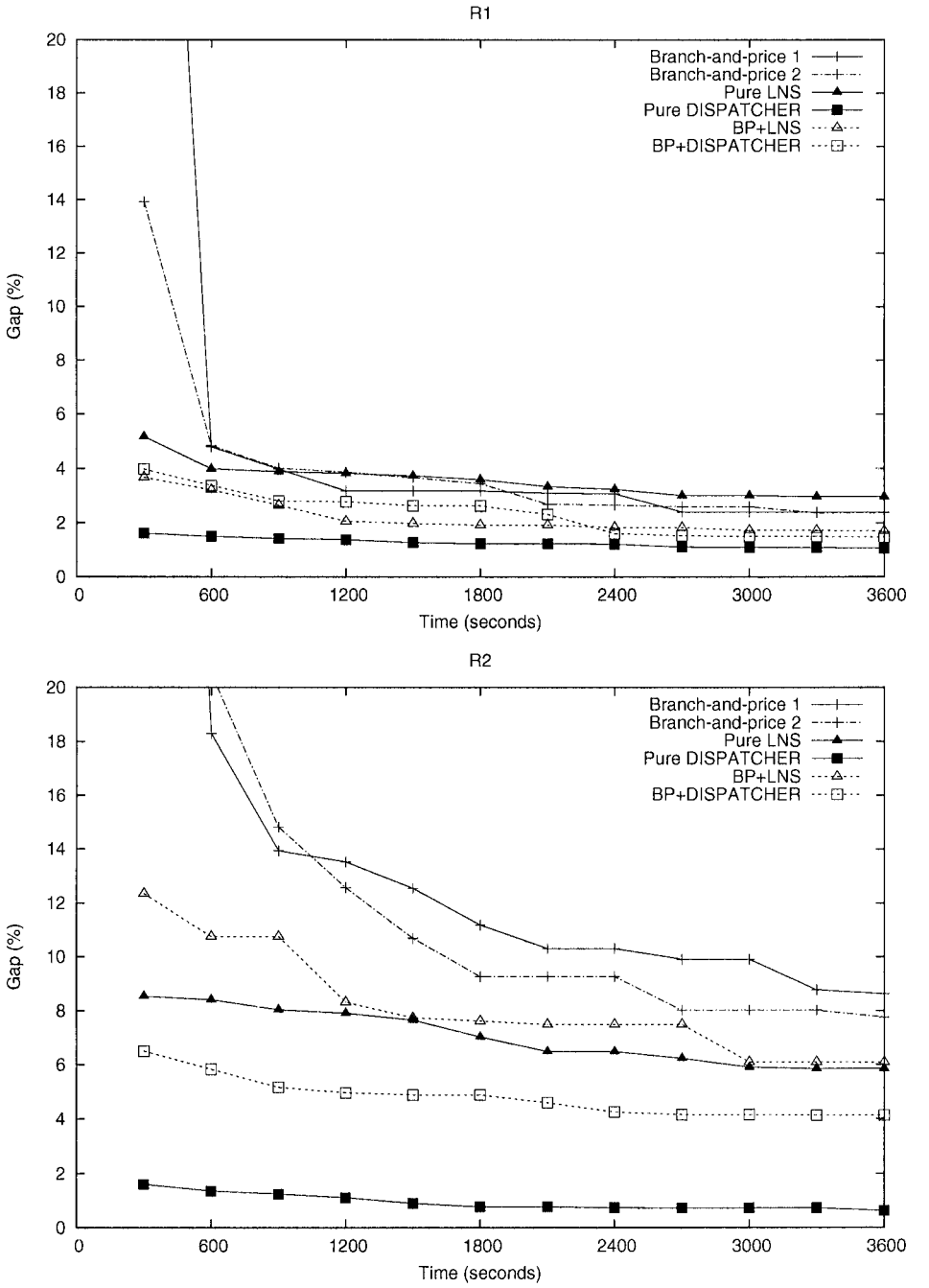

Figure 4.3. Evolution of solution quality over time for R series (100-customer instances).

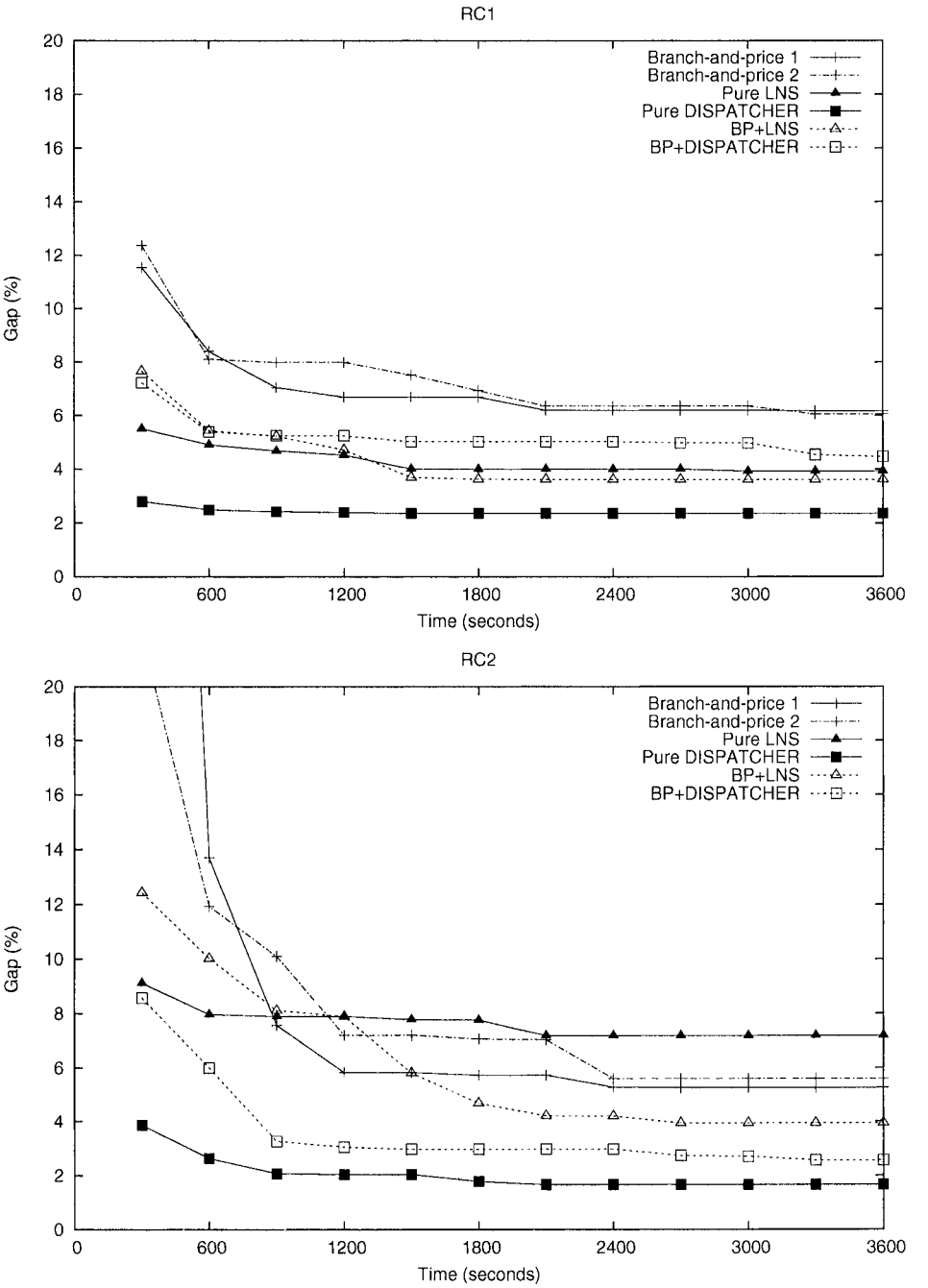

Figure 4.4. Evolution of solution quality over time for RC series (100-customer instances).

improve on the same algorithm starting from a trivial solution ("BP 1"). Our tentative explanation is that the simple *savings* heuristic is not powerful enough to foster a significant difference in performance.

Let us examine a few examples of the 100-customer instances for more specific remarks. On series R1, RC1 and most significantly RC2, our cooperation scheme between branch-and-price and LNS outperforms both pure LNS and the two variants of pure branch-and-price. This illustrates the useful interaction between the two components of the hybrid scheme, each optimizing the solutions found by the other component. On series RC2, branch-and-price outperforms pure LNS and our cooperation scheme performs nonetheless better than both. This shows the robustness of our cooperation scheme. Note that although BP+DISPATCHER significantly improves on pure branch-and-price BP 1 and BP 2, it does not give quite as good results as ILOG DISPATCHER used alone: ILOG DISPATCHER is especially effective at providing rapidly very good solutions for all series, hence it is difficult to outperform. But our cooperative scheme has nonetheless the advantage of additionally providing the user with a tight lower bound on the objective.

4.4 What component finds integer solutions in the cooperation scheme?

Let us now give more detailed results for each component of our hybrid scheme (branch-and-price and local search) so as to better understand why they succeed in finding good integer solutions early. Tables 4.4 and 4.5 show for each component of the cooperation scheme the number of times it succeeds in finding a new and better solution, divided by the number of times this component was called (column *%s* for success), or divided by the total number of integer solutions found by all components (column *%c* for contribution). Statistics are aggregated for 100-customer instances of Solomon series 1 and 2.

Table 4.4. Integer solutions found with the cooperation BP+LNS.

Component	Series 1		Series 2	
	%s	%c	%s	%c
Relaxed master problem integer	0.01	4.65	0.01	1.85
MIP	5.92	7.90	7.96	7.40
Multiple visits heuristic	62.96	7.90	80.00	7.40
Total LNS	27.22	79.53	29.03	83.33
... when optimizing a solution found by:				
Branch-and-price	85.00	23.72	64.70	20.37
LNS	21.12	55.81	24.63	62.96

Table 4.5. Integer solutions found with the cooperation BP+DISPATCHER.

Component	Series 1		Series 2	
	%s	%c	%s	%c
Relaxed master problem integer	0.04	11.81	0.09	7.14
MIP	5.40	7.27	3.75	2.38
Multiple visits heuristic	35.71	6.81	45.83	4.36
Total ILOG DISPATCHER	24.84	74.09	35.05	86.11
... when optimizing a solution found by:				
Branch-and-price	63.73	26.36	50.54	18.25
ILOG DISPATCHER	18.58	47.72	32.38	67.85

The "Multiple visits heuristic" line refers to the greedy heuristic transforming a solution of the set covering formulation into a set partitioning solution, as described at the end of Section 3.1. It is mostly useful at the beginning of the optimization process: Afterwards, the upper bound is too tight to allow for customers to be visited more than once. Both our local search algorithms work on a model where each customer is visited exactly once. Therefore, the multiple visits heuristic cannot improve solutions found by local search. The line "Local search... when optimizing a solution found by branch-and-price" refers to the results of local search starting from a solution found when the relaxed master problem was integer, or from a solution found by the MIP solver, or from the output of the multiple visits heuristic optimizing a solution found by the two former methods.

On all series, local search finds the majority of solutions: The good results of our cooperation scheme are naturally obtained first thanks to the great ability of local search to find good feasible solutions. The success rate of local search is much higher when starting from a solution found by branch-and-price than when starting from a solution found by local search itself. This illustrates the diversification mechanism: When branch-and-price finds a solution, it is far from the last local search local optimum, hence it is more likely to be improved by local search.

4.5 Quality of lower bounds

In this section, we compare the ability of each method to provide lower bounds and evaluate whether our cooperative algorithms retain the ability of branch-and-price to generate good lower bounds. Recall that local search algorithms do not provide lower bounds nor optimality proofs.

Table 4.6 gives the mean relative deviation between the lower bound obtained by each studied algorithm and the reference solutions of Tables 4.1 and 4.2 on the 100-customers instances of Solomon series 1 and

Table 4.6. Quality of lower bounds (100-customer instances).

Instances	Series 1	Series 2
BP 1	-0.76%	-0.34%
BP 2	-0.77%	-0.37%
BP+LNS	-0.77%	-0.34%
BP+DISPATCHER	-0.75%	-0.33%

series 2. Recall that, on series 1, reference solutions are either optimal solutions or possibly sub-optimal lower bounds, therefore Table 4.6 does not indicate for this series upper bounds on the distance between the lower bounds obtained and the optimum. Note also that on several instances (1 instance in series 1, 15 instances in series 2), none of the algorithms studied produces a lower bound on the objective: The one-hour time limit is too short to terminate pricing at root node. These instances are not taken into account for the computation of the mean relative deviation in Table 4.6. The overall conclusion of Table 4.6 is that both of our hybrids between branch-and-price and local search (LNS or ILOG DISPATCHER) retain the ability of branch-and-price to generate good lower bounds. Note also from Table 4.3 that our cooperation scheme between branch-and-price and local search produces approximately the same number of optimality proofs as pure branch-and-price.

4.6 Proof of optimality for previously unsolved instances

We finally present result for two previously open instances which we solved to optimality: R211 and RC204, both with 50 customers. Note that R211.50 was recently solved independently to optimality, as reported in Irnich and Villeneuve (2003). Table 4.7 gives for each instance the minimal distance and the corresponding number of vehicles. Table 4.8 gives the time in seconds needed to reach the optimal solution (T_{opt}) and subsequently prove optimality (T_{total}), and the number of nodes explored in the branch-and-price tree (nodes). We provide results for pure branch-and-price (BP 1 and BP 2) and for our cooperation scheme between branch-and-price and local search (BP+LNS and BP+DISPATCHER).

Note that BP+DISPATCHER is the only algorithm that solved instance RC204.50 to optimality. BP+LNS reached the optimal solution of RC204.50 but was not able to prove optimality within a week of CPU time. Pure branch-and-price (BP 1 and BP 2) also failed to solve

Table 4.7. Optimal values for two previously open instances.

Instance	R211.50	RC204.50
Cost	535.5	444.2
Number of vehicles	3	3

Table 4.8. Proof of optimality for two previously open instances.

Algorithm	R211.50			RC204.50		
	T_{opt}	T_{total}	nodes	T_{opt}	T_{total}	nodes
BP 1	115,100	196,868	257	-	-	-
BP 2	103,600	126,648	85	-	-	-
BP+LNS	214,100	300,184	281	152,100	-	-
BP+DISPATCHER	25,900	94,411	85	50,200	84,059	1

RC204.50 to optimality when given a week of CPU time. On this problem with a long horizon and large time windows, it appears to be extremely time consuming to compute interesting elementary constrained shortest paths. On the contrary, ILOG DISPATCHER succeeds in finding near-optimal routes within a reasonable time and branch-and-price can then find the optimal solution and prove optimality. This illustrates the fact that diversification for generating solutions but also individual columns is a key point of our cooperation scheme.

BP+DISPATCHER solves R211.50 to optimality faster than pure branch-and-price. The acceleration is especially visible when comparing the time needed to reach the optimal solution, but the total time including the optimality proof is also reduced. Note however that BP+LNS slows down the resolution and explores more nodes that BP 2. The phenomenon is witnessed for BP 1. Recall that our branching strategy is not fixed: The branching arc depends on the pool of columns already generated, this is why the number of nodes explored can vary from one branch-and-price variant to the next.

5. Conclusion

In this paper we introduced a new general strategy for combining local search and branch-and-price. We showed with extensive computational experiments on the vehicle routing problem with time windows that our cooperation scheme consistently improves the ability of pure branch-and-price to find good integer solutions early, while retaining the ability of branch-and-price to generate good lower bounds. The quality improvement of integer solutions generated is most often correlated with the effectiveness of the local search algorithm used, but significant im-

provements can be obtained even with a simple local search algorithm. It remains to be seen if our cooperation scheme will be applied successfully to different and more complex problems. We believe nonetheless that our results on a quite difficult problem and with two local search algorithms of varied effectiveness are encouraging.

Our cooperation scheme generalizes three previously known accelerations for branch-and-price and can be applied to any branch-and-price model. It can also be seen as the generalization to branch-and-price of the use of heuristics in branch-and-bound. However, unlike most interesting heuristics for branch-and-bound, it is not domain-independent: A specific local search algorithm tailored to the problem at hand has to be written each time a different branch-and-price model is to be solved. We believe nonetheless that it is a step toward extending existing advanced strategies from branch-and-bound to branch-and-price.

Acknowledgements The work presented in this paper was performed while the first author was a PhD student at ILOG.

We wish to thank the anonymous referee whose detailed comments helped us to significantly improve this chapter. We are also thankful to Jacques Desrosiers, Stefan Røpke, and Dominique Feillet for their helpful comments on a preliminary version of this chapter.

References

Aarts, E. and Lenstra, J. (1997). Local Search in Combinatorial Optimization. Wiley.

Alt, H., Guibas, L., Mehlhorn, K., Karp, R., and Wigderson, A. (1996). A method for obtaining randomized algorithms with small tail probabilities. *Algorithmica*, 16(4-5):543–547.

Balas, E. and Martin, C. (1980). Pivot and complement—A heuristic for $0-1$ programming. *Management Science*, 26(1):89–96.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329.

Bixby, R., Fenelon, M., Gu, Z., Rothberg, E., and Wunderling, R. (2000). MIP: Theory and pratice—Closing the gap. *System Modelling and Optimization: Methods, Theory, and Applications*, pp. 19–49, Kluwer Academic Publishers.

Bräysy, O. and Gendreau, M. (2003a). Vehicle routing with time windows, part I: Route construction and local search algorithms. *Technical*

Report, SINTEF Applied Mathematics, Department of Optimization, Oslo, Norway.

Bräysy, O. and Gendreau, M. (2003b). Vehicle routing with time windows, part II: Metaheuristics. *Technical Report*, SINTEF Applied Mathematics, Department of Optimization, Oslo, Norway.

Chabrier, A. (2003). Vehicle routing problem with elementary shortest path based column generation. Forthcoming in: *Computers and Operations Research.*

Chabrier, A., Danna, E., and Le Pape, C. (2002). Coopération entre génération de colonnes avec tournées sans cycle et recherche locale appliquée au routage de véhicules *Huitièmes Journées Nationales sur la résolution de Problèmes NP-Complets* (JNPC'2002), pp.83–97.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581.

Cook, W. and Rich, J. (1999). A parallel cutting-plane algorithm for the vehicle routing problem with time windows. *Technical Report* TR99-04, Department of Computational and Applied Mathematics, Rice University.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, *The Vehicle Routing Problem* (M., and Soumis, F. (2002). The VRP with time windows. In: Toth, P. and Vigo, D., eds.), pp. 157–193, SIAM Monographs on Discrete Mathematics and Applications.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936.

Croes, G. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6:791–812.

De Backer, B., Furnon, V., Shaw, P., Kilby, Ph., and Prosser, P. (2000). Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6:501–523.

Desaulniers, G., Desrosiers, J., and Solomon, M. (2002). Accelerating strategies for column generation methods in vehicle routing and crew scheduling problems. In: *Essays and Surveys in Metaheuristics* (C. Ribeiro and P. Hansen, eds.) , pp. 309–324, Kluwer Academic Publishers.

Desrochers, M. (1986). La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes. *Ph.D Thesis*, Université de Montréal, Canada.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354.

Desrosiers, J. and Lübbecke, M. (2004). A primer in column generation. *Les Cahiers du GERAD*, G-2004-02, HEC, Montréal, Canada.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

Gambardella, L., Taillard, E., and Agazzi, G. (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: *New Ideas in Optimization* (D. Corne, M. Dorigo, , and F. Glover, eds.), pp. 63–76, McGraw-Hill.

Glover, F. and Laguna, M. (1997). Tabu Search, Kluwer Academic Publishers.

Hadjar, A., Marcotte, O., and Soumis, F. (2001). A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Les Cahiers du GERAD*, G-2001-25, HEC, Montréal, Canada.

Homberger, J. and Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37:297–318.

Houck, D., Picard, J., Queyranne, M., and Vemuganti, R. (1980). The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch* 17:93-109.

ILOG, S.A. (2002). ILOG DISPATCHER 3.3 User's Manual.

Irnich, S. (2001). The shortest path problem with k-cycle elimination ($k \geq 3$): Improving a branch and price algorithm for the VRPTW. *Technical Report*, Lehr- und Forschungsgebiet Unternehmensforschung Rheinish-Westfälishe Technische Hochshule, Aachen, Germany.

Irnich, S. and Villeneuve, D. (2003). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *Les Cahiers du GERAD*, G-2003-55, HEC, Montréal, Canada.

Kallehauge, B., Larsen, J., and Madsen, O. (2001). Lagrangean duality and non-differentiable optimization applied on routing with time windows—Experimental results. *Technical Report IMM-TR*-2001-9, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Kohl, N., Desrosiers, J., Madsen, O.B.G., Solomon, M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 1(13):101–116.

Larsen, J. (1999). Parallelization of the vehicle routing problem with time windows. *Ph.D Thesis*, Informatics and Mathematical Modelling, Technical University of Denmark, DTU.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245-2269.

Or, I. (1976). Traveling salesman-type problems and their relation to the logistics of regional blood banking. *Ph.D Thesis*, Department of Industrial Engineering and Management Sciences, Northwestern University.

Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34:336–344.

Rochat, Y. and Taillard, E. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167.

Røpke, S. (2003). A General Heuristic for Vehicle Routing Problems. *International Workshop on Vehicle Routing and Multi-modal Transporation* (ROUTE'2003).

Rousseau, L.-M., Gendreau, M., and Pesant, G. (2002). Solving small VRPTWs with constraint programming based column generation. In: *Proceedings of the Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (CP-AI-OR'02) (N. Jussien and F. Laburthe, F.,eds.), pp. 333–344.

Savelsbergh, M. and Sol, M. (1998). Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Pro-*

gramming (CP'98), pp. 417–431. Forthcoming in: INFORMS Journal of Computing.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35:254–265.

Voß, S., Martello, S., Osman, I., and Roucairol, C. (1999). Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers.

Voudouris, C. (1997). Guided local search for combinatorial optimization problems. *Ph.D Thesis*, Department of Computer Science, University of Essex, Colchester, UK.

Xu, H., Chen, Z., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364.