Appendix A: Dynamic Program Relaxation Example

We give an example of a dynamic program relaxation that is not a state-space relaxation. First, we give the formal definition of a state-space relaxation. We will use a formal definition similar to the one in Christofides et al. (1981). Given (S_1, h_1, c_1, G_1), define a state-space relaxation to be (S_2, h_2, c_2, G_2) that meets the following requirements. First, $|S_2| < |S_1|$. Second, there exists a mapping function $\mu : S_1 \to S_2$ such that for each $s_2 \in S_1$, for all $(s_1, u) \in h_1^{-1}(\{s_2\})$, $(\mu(s_1), u) \in h_2^{-1}(\{\mu(s_2)\})$, where we define $h^{-1}(\{s_2\}) = \{(s_1, d) : h((s_1, d)) = s_2\}$ as the preimage of $s_2 \in S$, not to be confused with an inverse function. Third, for every $s_1 \in S_2$, $c_2(s_1, u) = \min_{\{s_3 \in S_1 | \mu(s_3)=u, \mu(h_1(s_3,u))=h_2(s_1,u)\}}\{c_1(s_3, u)\}$.

We give an example of when a dynamic program relaxation is not a state-space relaxation in Proposition 2, using data for an example CVRP instance in Figure 6. The costs are not relaxed in either dynamic program, so we only argue about the solution sets complying with the definitions.

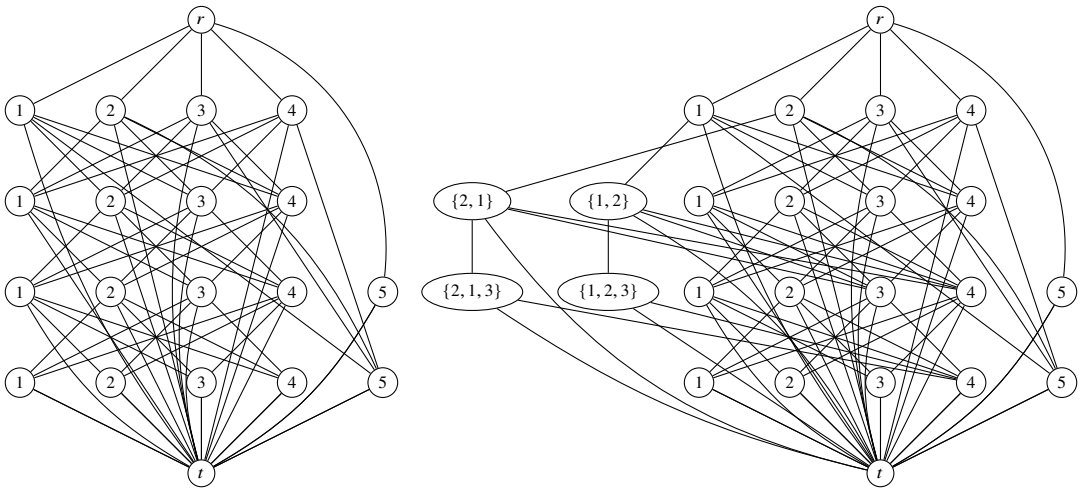
PROPOSITION 2. *The dynamic program relaxation represented by the state-transition graph on the right side of Figure 5 is not a state-space relaxation.*

Proof For sake of contradiction assume the dynamic program relaxation is a state-space relaxation defined by a mapping μ. It must be the case that $\mu(r) = r$, which implies $\mu((\{1\}, 1, 1)) = (\{1\}, 1, 1)$ and $\mu((\{2\}, 1, 2)) = (\{2\}, 1, 2)$. This implies $\mu((\{1, 2\}, 2, 2)) = (\{1, 2\}, 2, 2)$ and $\mu((\{2, 1\}, 2, 1)) = (\{1\}, 2, 1)$. Finally, this implies $\mu((\{1, 2, 3\}, 3, 3)) = (\{1, 2, 3\}, 3, 3)$, which would require $h((\{2, 1\}, 2, 1), 3) = (\{1, 2, 3\}, 3, 3)$, but in the dynamic program relaxation shown $h((\{2, 1\}, 2, 1), 3) = (\{2, 1, 3\}, 3, 3)$ (the states labelled $(\{1, 2, 3\}, 3, 3)$ and $(\{2, 1, 3\}, 3, 3)$ are distinct even though they represent the same information), which is a contradiction. $Q.E.D.$

Appendix B: Application Models

B.1. Vehicle Routing Problem with Time Windows (VRPTW)

The problem $\mathcal{P}$ has the same form as the CVRP, but the set $\mathcal{S}$ is further restricted to routes that respect the time window constraints. To create P, we extend the model for the CVRP as follows. We augment the states of the dynamic program to consider time. The states become tuples (NG, w, v, τ) where τ is the current time. The initial state is $r = (\varnothing, 0, 0, 0)$. Given a state $s = (\text{NG}, w, v, \tau)$ and element $u \in U$ such that $u \neq 0$, $u \notin \text{NG}$, $w + q_u \leq Q$, and $\tau + \ell_{v,u} \leq l_u$, the transition function becomes $h(s, u) = (\text{NG} \cup \{u\}, w + q_u, u, \max(\tau + \ell_{v,u}, e_u))$. Otherwise if $u = 0$ and $\tau + \ell_{v,0} \leq l_0$, define $h(s, 0) = t$. Otherwise $h(s, u) = -1$. The cost function is the same as for the CVRP. The constraints G are the same as for the CVRP, but without the constraint requiring K vehicles.



State-space relaxation. Dynamic program relaxation.

Figure 5 The state-transition graph on the left is a state-space relaxation of a dynamic program, and the state-transition graph on the right is a dynamic program relaxation created by removing some feasible sequences from the state-space relaxation on the left.

Locations $V = \{0, 1, 2, 3, 4, 5\}$
Depot = 0
Demands $q_1 = q_2 = q_3 = q_4 = 1$, $q_5 = 3$
Number of vehicles $K = 2$
Vehicle capacity $Q = 4$

l_{ij}	0	1	2	3	4	5
0	0	12	10	11	10	11
1	12	0	2	1	22	23
2	10	2	0	1	20	21
3	11	1	1	0	21	22
4	10	22	20	21	0	1
5	11	23	21	22	1	0

Figure 6 Input data for a CVRP instance.

We relax the model by creating a dynamic programming relaxation w.r.t. P. We use an ng-route relaxation and also consider relaxing the time windows and/or vehicle capacity. To relax the time windows, we use a 'bucketing' idea from the column generation literature (Sadykov et al. 2021). When a transition would create a state with time value τ, round down τ to the nearest multiple of $\Delta \in \mathbb{Z}$. To relax the load values, for certain instances we set all states to have load 0. This way, we only create states that remember load when conflicts are refined. We use a binary value κ to indicate if capacity should be relaxed or not. We add a counter c to all states to maintain an acyclic state-transition graph (Horn 2021). We use an upper bound denoted U on the number of locations in a route, because the relaxation can create many long infeasible routes. We calculate U by using two greedy methods based on summing the smallest loads up to Q and summing the shortest distances with service times compared to l_0.

Formally, we construct an initial dynamic programming relaxation $P' = (S', h', c', G')$ w.r.t. P. Each state is a tuple $(\text{NG}, w, v, \tau, c)$ with the initial state being $r_1 = (\emptyset, 0, 0, 0, 0)$. The set of states S'

is implicitly defined by defining the following transition function h'. Given a state $s = (\text{NG}, w, v, \tau, c)$ and label $u \in U$ such that $u \neq 0, u \notin \text{NG}, w + q_u \leq Q, \tau + \ell_{v,u} \leq l_u$, and $c < U$, let the transition function be $h'(s,u) = ((\text{NG} \cup \{u\}) \cap N_u, (w+q_u) * (1-\kappa), u, \max(\lfloor \frac{\tau+\ell_{v,u}}{\Delta} \rfloor * \Delta, e_v), c+1)$. Otherwise, when $u = 0$ and $\tau + \ell_{v,0} \leq l_0$, define $h'(s,0) = t$. Otherwise $h'(s,u) = -1$. Then, we keep the same cost function $c' = c$ and the same additional cost function $G' = G$.

B.2. Graph Multicoloring

The problem $\mathcal{P}$ is formed by an element set $U = \{1, ..., |V|\}$, a set $\mathcal{S}$ containing all independent sets in $\mathcal{G}$ as sequences with strictly increasing values, a constant cost function $f = 1$, and for a subset of sequences X, $C(X) = 1$ if and only if all vertices are contained in at least one sequence. We represent the constraint function as a tuple $(\gamma_j, =, b_j)$ for each $j = 1, ..., |V|$ such that $\gamma_j(x) = [\![j \in x]\!]$.

We model the problem with a dynamic program $P = (S, h, c, G)$. The dynamic program will depend on an ordering of the vertices $\{1, \ldots, |V|\}$, similar to the one in van Hoeve (2022). In our experiments, we use a variable ordering called 'min width' from Karahalios and van Hoeve (2022). Let each state $s = (\text{NG})$ contain a 'no-good' subset of vertices NG that can no longer be included in an independent set. The initial state is $r = (\varnothing)$. The transition function given a state $s = (\text{NG})$ and decision i such that $i \notin \text{NG}$ is $h(s,i) = (\text{NG} \cup N_i \cup \{1, \ldots, i\})$, where N_i is the set of neighbors of vertex i, and all vertices with smaller index are included in the updated NG to break symmetries. The dynamic program differs from the one in van Hoeve (2022), because in that work the set of decisions was binary and each transition corresponded to selecting a vertex to be in the independent set or not. The cost function is $c(r,i) = 1$ for all $i \in U$, and $c(s,i) = 0$ when $s \neq r$ for all $i \in U$. Second, we construct G. For each $j = 1, ..., m$, we create an additional cost function $g_j((s,u)) = [\![j = u]\!]$.

We relax the model by creating a dynamic programming relaxation $P' = (S', h', c', G')$ w.r.t. P. Each state maintains a 'no-good' subset of vertices $s = (\text{NG})$ and the initial state is $r = (\varnothing)$. The dynamic program only 'remembers' the latest decision. Formally, given a state $s = (\text{NG})$ and feasible transition $u \in U$ such that $u \notin \text{NG}$, define $h'(s,u) = (\{1, \ldots, u\} \cup N_u)$. Otherwise, $h'(s,u) = -1$. We keep the same cost function $c' = c$ and additional cost functions $G' = G$.

In our experiments, we use a common preprocessing rule to simplify the graph $\mathcal{G}$ before setting up the model. We remove a vertex if the sum of the weights of its neighbors plus its own weight is smaller than a lower bound on the optimal solution value. For multicoloring, we use an initial maximum (weighted) clique from Gualandi and Malucelli (2012) as an initial lower bound for this preprocessing.

B.3. Pickup-and-Delivery Problem with Time Windows (PDPTW) and Sequential Ordering Problem (SOP)

The problem $\mathcal{P}$ has the same form as the VRPTW, but the sequences in $\mathcal{S}$ must also follow the precedence constraints. We define the set of precedence locations for each location $u \in V$ as Π_u. So, for $u \in D$, $\Pi_u = \{u - n\}$, and $\Pi_u = \emptyset$ otherwise. We define the transition function as follows. Each state is a tuple $s = (\text{NG}, w, v, \tau)$ and the initial state is $r = (\text{NG}, w, v, \tau)$. Given a state $s = (\text{NG}, w, v, \tau)$ and element $u \in U$ such that $u \neq 0$, $u \notin \text{NG}$, $\Pi_u \subseteq \text{NG}$, $w + q_u \leq Q$, and $\tau + \ell_{v,u} \leq l_u$, let $h(s,u) = (\text{NG} \cup \{u\}, (w + q_u) * (1 - \kappa), u, \lfloor \frac{\max(\tau + \ell_{v,u}, e_u)}{\Delta} \rfloor * \Delta, c + 1)$. Otherwise, when $u = 0$ and $\tau + \ell_{v,0} \leq l_0$, define $h(s,0) = t$. Otherwise $h(s,u) = -1$. The cost function and the additional costs are the same as for the VRPTW.

We relax the model in a similar way to the VRPTW, while also relaxing precedence constraints. To relax the model, we create a dynamic programming relaxation $P' = (S', h', c', G')$ w.r.t. P. Again, we add a counter c to each state, so each state has the form $s = (\text{NG}, w, v, \tau, c)$. The initial state is $r = (\emptyset, 0, 0, 0, 0)$. Then, for a state $s = (\text{NG}, w, v, \tau, c)$ and transition u such that $u \neq 0$, $u \notin \text{NG}$, $w + q_u \leq Q$, $\tau + \ell_{v,u} \leq l_u$, and $c < |V|$, let $h'(s,u) = ((\text{NG} \cup \{u\}) \cap N_u, (w + q_u) * (1 - \kappa), u, \max(\lfloor \frac{\tau + \ell_{v,u}}{\Delta} \rfloor * \Delta, e_u), c + 1)$. When $u = 0$, let $h'(s,u) = t$. Otherwise $h'(s,u) = -1$. We use the same cost function $c_1 = c$ and additional cost functions $G' = G$.

Similar to other works in the vehicle routing literature, we aim to solve instances based on distances that are rounded to the thousandths place. To do this, we start by rounding distances to the hundredths place and solving the instance. Then, we keep the P_i at termination, update the distances to be rounded to the thousandths place, and solve the instance again. We use best-known solution values from the literature, which sometimes use different rounding, which accounts for small discrepancies in the full table of results in Appendix N. We model the SOP in this way, with the additional constraint that a solution has one sequence.

Appendix C: Capacitated Vehicle Routing Problem (CVRP) Formulation

The model F to solve the CVRP is formulated as follows:

$$F: \min \sum_{a \in \mathcal{A}} c_a y_a \tag{9}$$

$$\text{s.t.} \sum_{a \in \delta^+(r)} y_a = K \tag{10}$$

$$\sum_{(s,u) \in \mathcal{A}: u=j} y_a = 1 \qquad \forall j = 1, ..., m \tag{11}$$

$$\sum_{a \in \delta^+(u)} y_a - \sum_{a \in \delta^-(u)} y_a = 0 \qquad \forall u \in \mathcal{N} \tag{12}$$

$$y_a \in \{0, 1\} \qquad \forall a \in \mathcal{A}. \tag{13}$$

The objective function (9) minimizes the sum of all arc costs. The 'flow conservation' constraints (12) ensure that the solution is a collection of labeled r-t paths, or single feasible routes. Constraint (10) enforces that exactly K units of flow originate from r and thus K routes are used. Constraints (11) ensure that all locations are visited once. The binary constraints (13) complete the formulation.

Appendix D: Variable Fixing

Variable fixing is an important method to improve the performance of column elimination. Variable fixing is an acceleration method used in integer programming (Nemhauser and Wolsey 1988) and constraint programming (Focacci et al. 1999) to prove that a variable must equal its lower bound or upper bound in an optimal solution. For integer programming, the proof requires a primal bound and a feasible dual solution to the linear programming relaxation. For column elimination, we use a variable fixing algorithm that considers one arc $a \in \mathcal{A}$ at a time and reasons about all $r-t$ paths that traverse the arc. Theorem 6 generalizes the arc fixing theorem from Karahalios and van Hoeve (2023), which is based on prior works (Irnich et al. 2010, Tanaka et al. 2009, Pessoa et al. 2010).

Let ν be a feasible solution to the dual of $LP(F)$ such that each ν corresponds to $\mathcal{G}$. For each arc $a \in \mathcal{A}$ we define a 'reduced cost distance' $rc(y_a) = c_a - \sum_{j=1}^{m} g_j(a)\nu_j$. For each node $u \in \mathcal{N}$, we define $sp_u^{\downarrow}$ as the shortest r-u path in the state-transition graph of P with respect to the reduced cost distances, and similarly define $sp_u^{\uparrow}$ to be the shortest u-t path in the state-transition graph of P.

THEOREM 6. *Consider arc $a = (v_1, v_2) \in \mathcal{A}$. Let $v(\nu)$ be the solution value of ν to the dual of $LP(F)$, and let UB an upper bound on the optimal solution value for F. If $v(\nu) + sp_{v_1}^{\downarrow} + sp_{v_2}^{\uparrow} + rc(y_a) > UB$, then arc a can be fixed to have flow 0 in F.*

Proof Consider the following integer program that is equivalent to F, created by enumerating the solutions to $\mathcal{S}$, where $\mathcal{A}_x$ is the set of arcs in the solution x.

$$(IP) \quad \min \sum_{x \in \mathcal{X}} z_x f(x) \tag{14}$$

$$\text{s.t.} \sum_{x \in \mathcal{X}} z_x \sum_{a \in \mathcal{A}_x} g_j(a) \circ_j b_j \qquad \forall\, j \in \{1, \ldots, m\} \tag{15}$$

$$z \in \mathbb{Z}_+^{|\mathcal{X}|} \tag{16}$$

Given ν and a solution x, in the linear program relaxation of IP, the variable z_x has reduced cost $rc(x) = f(x) - \sum_{j=1}^{m} \nu_j \sum_{a \in \mathcal{A}_x} g_j(a)$. Each x corresponds to a path $p = \{a_1, \ldots, a_l\}$ in $\mathcal{D}$, so $rc(x)$ can be decomposed into $rc(x) = \sum_{a \in \mathcal{A}_x} rc(y_a)$. For all p that contain arc a, let p' be the path that corresponds to the route x with lowest reduced cost. Denote the lowest reduced cost as $rc'(x') = sp^{\downarrow}_{v_1} + sp^{\uparrow}_{v_2} + rc(y_a)$. Now for sake of contradiction assume an optimal solution to F has $y_a = 1$. Then some solution x'' in $\mathcal{D}$ that contains arc a will be in the solution X to F, which is equivalent to $z_{x''} = 1$ in IP. To construct the remainder of an optimal solution to the linear programming relaxation of IP, we can solve the linear programming relaxation with the constraints defined by G adjusted to have the values contributed from x'' removed. Then, ν remains feasible to the dual of this updated problem and has value $v(\nu) - \sum_{j=1}^{m} \nu_j \sum_{a \in \mathcal{A}_{x''}} g_j(a)$. So, combining this feasible dual with the cost of x'' gives a valid lower bound on IP as $v(\nu) - \sum_{j=1}^{m} \nu_j \sum_{a \in \mathcal{A}_{x''}} g_j(a) + f(x'')$. This contradicts UB. Specifically, $v(\nu) - \sum_{j=1}^{m} \nu_j \sum_{a \in \mathcal{A}_{x''}} g_j(a) + f(x'') = v(\nu) + rc(x'') \geq v(\nu) + rc(x') \geq v(\nu) + sp^{\downarrow}_{v_1} + sp^{\uparrow}_{v_2} + rc(y_a) > UB$. Q.E.D.

Appendix E: Convergence of Column Elimination with Subgradient Descent

It is not straightforward to analyze the convergence of column elimination with subgradient descent. Given F_i, subgradient descent will converge to an optimal dual solution for $LP(F_i)$ if a divergent series step-length is used (Anstreicher and Wolsey 2009). However, an optimal primal solution is needed to determine if any conflicts need to be refined. Subgradient descent produces a sequence of solutions to the Lagrangian relaxation whose average converges to an optimal primal solution, but this does not theoretically guarantee that an optimal primal solution is found (Anstreicher and Wolsey 2009).

So, to evaluate the convergence of the algorithm in practice, we consider the following example. We apply column elimination and column elimination with subgradient descent to solve the instance C1_2_5 of the Vehicle Routing Problem with Time Windows (VRPTW) from Gehring and Homberger (2002), which we define in the next section. For each algorithm, we plot the sequence of dual solutions obtained at each iteration, converted into three dimensions using a principal component analysis method. Figure 7(a) shows that the optimal dual solutions of column elimination can greatly change from one iteration to the next, possibly indicating degeneracy, which can hinder column generation. In contrast, Figure 7(b) shows a smooth path of dual solution values, indicating that each step of subgradient descent is in the general direction of the optimal dual solution to $LP(F)$ that the algorithm finds upon termination.



a. CPLEX b. Lagrangian Method

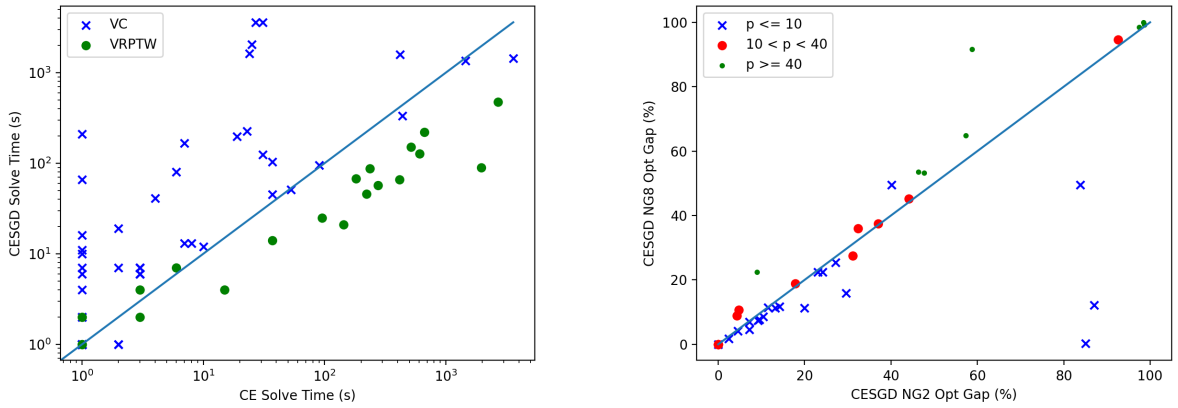
Figure 7 Sequences of dual solutions obtained by running a column elimination and column elimination with subgradient descent to solve the VRPTW instance C1_2_5. The dual solutions are plotted in three dimensions using the Python package 'sklearn'.

Appendix F: Evaluating Column Elimination with Subgradient Descent

We give insights into when column elimination with subgradient descent will outperform column elimination. To do this, we apply each algorithm to solve the vertex coloring instances from Johnson and Trick (1996) and the VRPTW instances from Gehring and Homberger (2002) and Solomon (1987). We choose these two applications, because they differ in the following way. The primal and dual solutions to F_i and F_{i+1} for some iteration i can be greatly different for vertex coloring. However, for the VRPTW, the primal and dual solutions can remain very similar. This property affects the performance of column elimination with subgradient descent for two reasons. Firstly, it changes the likelihood that refinements of conflicts at the current iteration will benefit future iterations. Secondly, the steps of subgradient descent are more likely to be in the direction of the

optimal solutions to F. So, we hypothesize that column elimination with subgradient descent will perform well for the VRPTW, but not for vertex coloring problem.

We show a plot that validates our hypothesis. For instances that both column elimination and column elimination with subgradient descent solver, we plot the time it takes for column elimination and column elimination with subgradient descent to solve each instance in Figure 8(a). For vertex coloring, column elimination solves instances on average 195 seconds faster than column elimination with subgradient descent. For the VRPTW, the column elimination with subgradient descent solves instances on average 330 seconds faster than column elimination. For vertex coloring, column elimination solves 13 additional instances and column elimination with subgradient descent solves no additional instances. For the VRPTW, column elimination with subgradient descent solves two additional instances and column elimination solves one additional instance.



a. Subgradient Descent b. Initial Relaxations for SOP

Figure 8 a) The time taken to solve VRPTW and vertex coloring (VC) instances using column elimination (CE) and column elimination with subgradient descent (CESGD). The axes use log scales. b) The optimality gap achieved when using column elimination with subgradient descent starting with an initial ng-route relaxation with $\rho = 2$ (CESGD NG2) and an initial ng-route relaxation with $\rho = 8$ (CESGD NG8) for the SOP with p% precedence constraints.

Appendix G: Sensitivity to Initial Relaxations

We evaluate the sensitivity of column elimination to the initial relaxation. To do this, we use column elimination with subgradient descent and two different initial relaxations to solve the TSPLIB SOP instances (Ascheuer et al. 2000). The two initial relaxations are ng-route with $\rho = 2$ and $\rho = 8$. We aim to test the behavior of the following general tradeoff. A stronger initial relaxation can reduce the number of conflict refinements needed for column elimination to solve the problem, but it can

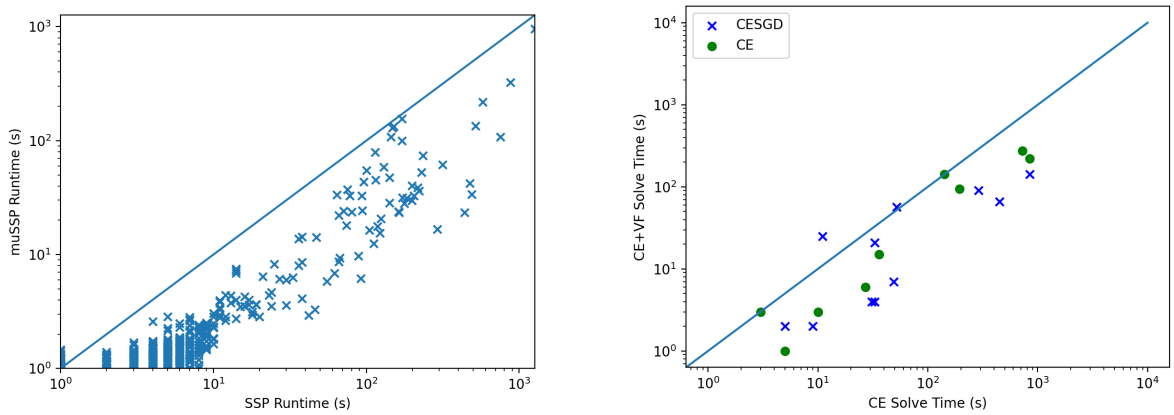
increase the time needed to solve $LP(F_i)$ at each iteration. In the context of the SOP, we consider that an initial relaxation relaxes precedence constraints and the constraint that the solution is an elementary path. We hypothesize that a stronger ng-route relaxation will capture the constraint that a solution is an elementary path, but not capture the precedence constraints because the NG sets are different for each location, so larger NG sets do not necessarily encode precedence constraints for solutions that have two locations appearing far apart in the ordering.

We show a plot that gives insight into this hypothesis. We plot the optimality gaps achieved at termination when starting with the $\rho = 2$ and $\rho = 8$ initial relaxations in Figure 8(b). The instances are partitioned based on the percent of values in the distance matrix that indicate a precedence, which we denote at p. The groups are $p \leq 10$, $10 < p < 40$, and $p \geq 40$. The instances in the group $p \leq 10$ tend to have a smaller number of vertices than the instances in the group with $p \geq 40$. The average number of locations is 63 and 263 respectively. The plot shows that for instances with a low percent of precedences, starting with the $\rho = 8$ initial relaxation is beneficial. For these instances, the size of the initial relaxation for $\rho = 8$ is small enough that column elimination can run for many iterations; when using $\rho = 8$ the median number of iterations solved is 3507 and when using $\rho = 2$ the median is 19389 iterations. In comparison, the plot shows that for instances with a high percent of precedences, starting with the $\rho = 2$ initial relaxation is beneficial. For these instances, when using $\rho = 8$, the median number of iterations solved is only 206, and when using $\rho = 2$ the median is 1038 iterations.

We give a full table of results with a comparison to a state-of-the-art solver in Appendix O. For high level statistics, column elimination solved 6 instances to optimality, failed to compute a lower bound for 8 instances, and returned an average optimality gap of 36.3% for the remaining instances. In comparison, the state-of-the-art solver (Libralesso et al. 2020) solved 8 instances to optimality and returned an average optimality gap of 87% for the remaining instances.

Appendix H: Impact of Using Variable Fixing

We evaluate the effect of using variable fixing during column elimination. We use column elimination with and without subgradient descent, and with and without variable fixing, to solve the VRPTW instances from Solomon (1987). We plot the run times of instances solved by both methods in Figure 9(a). Variable fixing allows column elimination with subgradient descent to solve one additional instance and solved instances on average 106 seconds faster than without variable fixing. Variable fixing allows column elimination without subgradient descent to solve 4 additional



a. Impact of muSSP b. Impact of Variable Fixing

Figure 9 a) The difference in solve time of $L(\lambda)$ between SSP and muSSP for solving VRPTW instances with column elimination. For both plots, the axes use log scales. b) The runtime to solve CVRP instances using column elimination and column elimination with subgradient descent both with and without variable fixing.

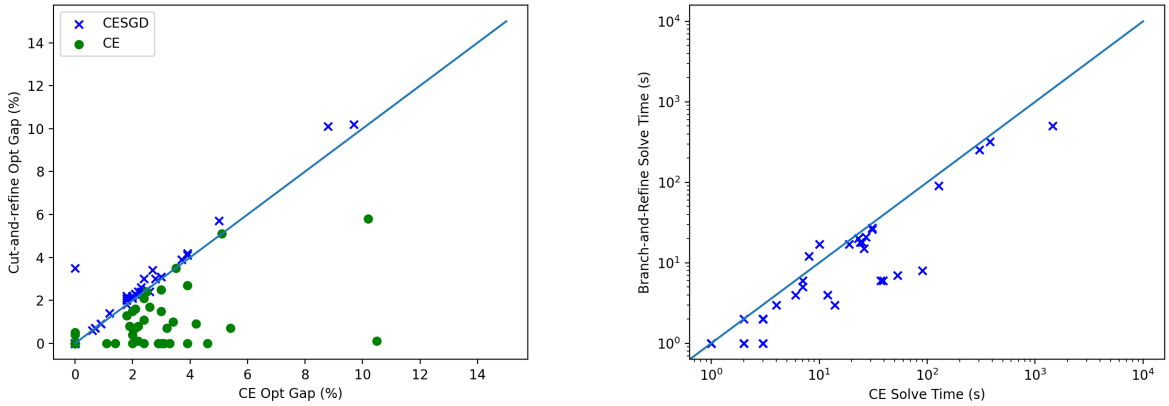
instances and solved instances on average 110 seconds faster than without variable fixing. These are the first experimental results for using variable fixing during column elimination with subgradient descent by checking dual feasibility and repairing infeasible duals.

Appendix I: Impact of Using Minimum Update SSP

We evaluate the impact of the minimum update successive shortest paths (muSSP) instead of SSP during column elimination with subgradient descent. We use column elimination with subgradient descent to solve ten CVRP instances from Uchoa et al. (2017) each with a different number of locations. At each iteration, we solve $L(\lambda)$ using both muSSP and SSP. We plot the runtimes in Figure 9(b). For smaller instances, there is not much impact, but for larger instances muSSP can greatly improve performance. Using muSSP solves the subproblem on average 3.7 times faster than using SSP.

Appendix J: Evaluating Cut-and-refine

We evaluate the performance of cut-and-refine compared to column elimination. We implement cut-and-refine for the CVRP using the framework in Figure 3. We apply column elimination without adding any cutting planes and cut-and-refine on the CVRP instances from `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/` in classes A,B,M,E,F, and P. We also apply column elimination with subgradient descent without any cutting planes and cut-and-refine with subgradient descent to the same instances. We choose to remove the constraint that a solution must use K



a. Cut-and-refine b. Branch-and-refine

Figure 10 a) The runtime to solve VRPTW instances using column elimination with and without subgradient descent, with and without cuts. b) The runtime to solve vertex coloring instances using column elimination (CE) and branch-and-refine. For this plot the axes use log scales.

vehicles, as this is the case for the large-scale VRPTW instances that we will evaluate when comparing to the state-of-the-art. Similar experiments that include the constraint on the number of vehicles are shown in Karahalios and van Hoeve (2023).

We plot the optimality gaps achieved at termination for both column elimination without cuts, column elimination with subgradient descent without cuts, cut-and-refine, and cut-and-refine with subgradient descent in Figure 10(a). The plot shows that cut-and-refine achieves better optimality gaps than column elimination without cutting planes. However, the plot also shows that cut-and-refine with subgradient descent does not improve the performance of column elimination with subgradient descent without cutting planes.

The cuts that we implement are categorized as ‘robust’ in the column generation literature. This name comes from the fact that they do not ‘complicate’ the pricing problem. Much work has been done in the column generation literature to incorporate ‘non-robust’ cuts, which do complicate the pricing problem (Pecin et al. 2017). For many instances of vehicle routing problems, the addition of non-robust cuts called subset-row inequalities is important to closing the instances, which currently gives these methods an advantage over column elimination. Non-robust cuts have not been effectively added in column elimination, although ideas similar to the one in Hoogendoorn and Dalmeijer (2025) may work.

Appendix K: Evaluating Branch-and-refine

We evaluate the performance of branch-and-refine. We implement branch-and-refine for the vertex coloring problem. We use Zykov branching, which chooses two nonadjacent vertices and defines one branch by contracting these vertices and the other branch by adding an edge between the vertices (Corneil and Graham 1973). We choose the two nonadjacent vertices with the highest sum of their degrees, using an ordering of the vertices as a tie breaker. We terminate the algorithm when solving a subproblem if there has not been an improvement to the lower bound for 30 seconds. We choose the subproblem with the greatest lower bound as the one to solve next, using the most recently created node as a tie breaker. We solve the DIMACS set of vertex coloring instances using branch-and-refine and column elimination without branch-and-bound (Johnson and Trick 1996).

We plot the runtimes of column elimination and branch-and-refine on instances that both methods solve in Figure 10(b). Branch-and-refine solves instances on average 2.3 times faster than without using branching. Branch-and-refine solves an additional 7 instances, mostly FullIns instances related to Mycielski graphs. Using column elimination without branching solves an additional 8 instances, mostly larger instances that require more conflict refinements.

We also compare to a state-of-the-art solver, given the same timeouts and initial upper bounds, and run the instances on the same server Held et al. (2011). Column elimination solves 67 instances to optimality, fails to give a bound on 9 instances, and has an average gap of 33.7% on the remaining instances. The state-of-the-art solver solves 89 instances to optimality, fails to give a bound on 18 instances, and has an average gap of 15.6% on the remaining instances. A full table of results is given in Appendix P.

Appendix L: Vehicle Routing Problem with Time Windows Results

In Table 6 we compare column elimination with VRPSolver on the vehicle routing problem with time windows (VRPTW). For each instance, we list the current best-known upper bound (UB) that each algorithm takes as input. For VRPSolver we report the lower bound (LB), the number of explored branch-and-cut-and-price nodes (Nodes), and the solution time (Time (s)). For column elimination, we report the lower bound (LB), the number of column elimination iterations (CEIt) and column elimination with subgradient descent iterations (CESIt), the number of cuts (Cuts), the number of refinements (CR), and the solution time (Time (s)).

Table 6 A comparison of the performance of VRPSolver from Pessoa et al. (2020) and column elimination for solving VRPTW instances by Gehring and Homberger (2002).

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CEIt	CESIt	CR	Cuts	Time (s)
C1_10_1	42444.8	42444.8	1	1219	42444.8	4	220	576	0	756
C1_10_10	39816.8	-	-	3600	38158.9	1	89	13090	0	3600
C1_10_2	41337.8	41068.76	1	3600	40078.2	1	154	9126	0	3600
C1_10_3	40064.4	-	-	3600	38245.5	1	25	8468	0	3600
C1_10_4	39434.1	-	-	3600	37825.6	1	19	4699	0	3600
C1_10_5	42434.8	42434.8	1	1227	42355.9	1	423	5405	0	3600
C1_10_6	42437	42437.0	1	1670	41987.5	1	323	12458	0	3600
C1_10_7	42420.4	42305.87	1	3600	41673.7	1	472	13079	0	3600
C1_10_8	41652.1	41062.0	1	3600	39687.8	1	292	19422	0	3600
C1_10_9	40288.4	39508.04	1	3600	38459.2	1	144	15855	0	3600
C1_2_1	2698.6	2698.6	1	11	2698.6	3	10	7	0	1
C1_2_10	2624.7	2624.7	1	218	2519.6	6	1009	12510	0	3600
C1_2_2	2694.3	2694.3	1	29	2681.66	21	2610	7624	0	3600
C1_2_3	2675.8	2675.8	3	338	2610.87	3	871	8836	0	3600
C1_2_4	2625.6	2625.6	1	457	2519.19	1	540	9155	0	3600
C1_2_5	2694.9	2694.9	1	16	2694.9	3	30	89	0	7
C1_2_6	2694.9	2694.9	1	20	2694.9	3	40	225	0	15
C1_2_7	2694.9	2694.9	1	18	2694.9	4	50	220	0	22
C1_2_8	2684	2684.0	1	26	2657.08	21	2617	8763	0	3600
C1_2_9	2639.6	2639.6	1	65	2568.08	10	1382	13212	0	3600
C1_4_1	7138.8	7138.8	1	99	7138.8	3	20	26	0	11
C1_4_10	6825.4	6820.19	1	3600	6631.87	1	365	10586	0	3600
C1_4_2	7113.3	7113.3	7	587	7060.35	2	591	4926	0	3600
C1_4_3	6929.9	6929.9	1	991	6797.26	1	251	7320	0	3600
C1_4_4	6777.7	6769.16	1	3600	6589.6	1	129	9037	0	3600
C1_4_5	7138.8	7138.8	1	134	7138.8	3	50	296	0	63
C1_4_6	7140.1	7140.1	1	204	7140.1	4	90	1066	0	202
C1_4_7	7136.2	7136.2	1	185	7136.2	51	120	1987	0	1959
C1_4_8	7083	7083.0	5	1128	6941.63	1	641	9803	0	3600
C1_4_9	6927.8	6927.8	1	1547	6725.5	1	428	10911	0	3600
C1_6_1	14076.6	14076.6	1	292	14076.6	3	50	75	0	55
C1_6_10	13617.5	13520.25	1	3600	13188.1	1	187	11778	0	3600
C1_6_2	13948.3	13948.3	15	1616	13775.7	1	269	4096	0	3600
C1_6_3	13757	13702.24	1	3600	13368.6	1	110	8562	0	3600
C1_6_4	13538.6	13347.86	1	3600	13090.8	1	64	7567	0	3600
C1_6_5	14066.8	14066.8	1	393	14066.8	3	150	797	0	462
C1_6_6	14070.9	14070.9	1	531	14054.8	11	390	3303	0	3600
C1_6_7	14066.8	14066.8	1	476	14058.1	15	320	2860	0	3600
C1_6_8	13991.2	13967.03	3	3600	13689.5	1	462	10889	0	3600
C1_6_9	13664.5	13649.77	1	3600	13299.7	1	287	12421	0	3600
C1_8_1	25156.9	25156.9	1	761	25156.9	3	80	222	0	187
C1_8_10	24026.7	23640.28	1	3600	23129.2	1	141	12807	0	3600
C1_8_2	24974.1	24910.3	1	3600	24397.0	1	211	7703	0	3600
C1_8_3	24156.1	23865.67	1	3600	23373.3	1	77	9104	0	3600
C1_8_4	23797.3	-	-	3600	22809.0	1	23	6233	0	3600
C1_8_5	25138.6	25138.6	1	737	25138.6	3	210	2012	0	1421

Table 6 Continued

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CEIt	CESIt	CR	Cuts	Time (s)
C1_8_6	25133.3	25133.3	1	1056	25030.9	1	373	5943	0	3600
C1_8_7	25127.3	25127.3	7	1140	24965.2	1	398	5693	0	3600
C1_8_8	24809.7	24688.04	1	3600	24050.7	1	409	18644	0	3600
C1_8_9	24200.4	23972.45	1	3600	23346.5	1	218	14876	0	3600
C2_10_1	16841.1	-	-	3600	16802.5	2	100	1012	0	3600
C2_10_10	15728.6	-	-	3600	12993.3	1	48	1575	0	3600
C2_10_2	16462.6	-	-	3600	15127.1	1	74	2069	0	3600
C2_10_5	16521.3	-	-	3600	15639.4	1	197	6448	0	3600
C2_10_6	16290.7	-	-	3600	15134.9	1	173	5396	0	3600
C2_10_7	16378.4	-	-	3600	15056.9	1	112	3550	0	3600
C2_10_8	16029.1	-	-	3600	14402.2	1	90	2615	0	3600
C2_10_9	16075.4	-	-	3600	12279.1	1	35	1490	0	3600
C2_2_1	1922.1	1922.1	1	228	1922.1	4	70	24	0	37
C2_2_10	1791.2	1791.2	1	631	1682.31	1	584	4204	0	3600
C2_2_2	1851.4	1851.4	1	387	1821.94	2	604	3440	0	3600
C2_2_3	1763.4	1753.63	3	3600	1661.01	1	259	2062	0	3600
C2_2_4	1695	1666.49	3	3600	1507.88	1	164	1443	0	3600
C2_2_5	1869.6	1869.6	1	276	1837.66	6	1069	7902	0	3600
C2_2_6	1844.8	1844.8	1	249	1784.65	1	799	5883	0	3600
C2_2_7	1842.2	1842.2	1	170	1788.26	1	677	5055	0	3600
C2_2_8	1813.7	1813.7	1	222	1731.34	1	692	4939	0	3600
C2_2_9	1815	1815.0	1	511	1728.5	1	590	4237	0	3600
C2_4_1	4100.3	4100.3	1	852	4087.63	17	310	1384	0	3600
C2_4_10	3665.1	3647.88	1	3600	3390.76	1	185	2479	0	3600
C2_4_2	3914.1	3900.22	1	3600	3804.51	1	170	2574	0	3600
C2_4_3	3755.2	3723.96	1	3600	3325.44	1	96	1468	0	3600
C2_4_4	3523.7	3486.12	1	3600	2718.37	1	70	748	0	3600
C2_4_5	3923.2	3923.2	1	971	3827.97	1	408	5653	0	3600
C2_4_6	3860.1	3860.1	1	2466	3699.51	1	309	4455	0	3600
C2_4_7	3870.9	3870.9	1	1483	3699.63	1	313	4385	0	3600
C2_4_8	3773.7	3770.24	1	3600	3557.85	1	265	3680	0	3600
C2_4_9	3842.1	3806.45	1	3600	3581.76	1	276	3671	0	3600
C2_6_1	7752.2	7719.46	1	3600	7691.56	3	541	1995	0	3600
C2_6_10	7123.9	6340.81	1	3600	**6526.6**	1	162	3253	0	3600
C2_6_2	7471.5	7075.15	1	3600	**7210.15**	1	130	2244	0	3600
C2_6_3	7215	4670.06	1	3600	**6181.3**	1	63	1072	0	3600
C2_6_5	7553.8	7540.44	1	3600	7269.61	1	304	6003	0	3600
C2_6_6	7449.8	7400.61	1	3600	7020.32	1	269	5470	0	3600
C2_6_7	7491.3	6294.69	1	3600	**7014.31**	1	258	5282	0	3600
C2_6_8	7303.7	7223.09	1	3600	6811.1	1	248	5038	0	3600
C2_6_9	7303.2	5754.15	1	3600	**6841.19**	1	195	3972	0	3600
C2_8_1	11631.9	-	-	3600	11616.6	9	150	1023	0	3600
C2_8_10	10946	-	-	3600	9898.34	1	133	2961	0	3600
C2_8_2	11394.5	-	-	3600	10777.8	1	108	3089	0	3600
C2_8_3	11138.1	-	-	3600	2920.84	1	13	1492	0	3600
C2_8_5	11395.6	-	-	3600	10911.9	1	248	6313	0	3600
C2_8_6	11316.3	-	-	3600	10624.8	1	248	6295	0	3600
C2_8_7	11332.9	-	-	3600	10613.4	1	185	4694	0	3600
C2_8_8	11133.9	-	-	3600	10280.1	1	184	4548	0	3600
C2_8_9	11140.4	-	-	3600	10242.4	1	147	3634	0	3600

Table 6 Continued

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CEIt	CESIt	CR	Cuts	Time (s)
R1_10_1	53046.5	52756.54	1	3600	51447.2	1	91	59	0	3600
R1_10_10	47364.6	46676.18	1	3600	-	-	-	-	-	3600
R1_10_5	50406.7	49928.13	1	3600	48279.2	1	34	269	0	3600
R1_10_9	49162.8	48632.73	1	3600	34575.0	1	1	0	0	3600
R1_2_1	4667.2	4667.2	1	16	4645.66	15	1731	195	0	3600
R1_2_10	3293.1	3285.31	15	3600	3150.43	1	98	1424	0	3600
R1_2_2	3919.9	3919.9	1	45	3688.7	1	99	1008	0	3600
R1_2_3	3373.9	3358.72	5	3600	2932.78	1	29	304	0	3600
R1_2_4	3047.6	3039.51	3	3600	-	-	-	-	-	3600
R1_2_5	4053.2	4053.2	3	399	3989.12	1	530	1959	0	3600
R1_2_6	3559.1	3552.78	13	3600	3328.57	1	82	1245	0	3600
R1_2_7	3141.9	3141.9	1	750	2892.17	1	33	481	0	3600
R1_2_8	2938.4	2938.4	5	2030	2699.9	1	20	249	0	3600
R1_2_9	3734.7	3734.7	3	731	3627.03	1	252	2017	0	3600
R1_4_1	10305.8	10305.8	3	338	10191.2	1	439	149	0	3600
R1_4_10	8077.8	8028.88	1	3600	7495.14	1	34	1031	0	3600
R1_4_2	8873.3	8843.47	7	3600	7759.51	1	26	609	0	3600
R1_4_3	7784.3	7698.59	9	3600	5194.35	1	1	0	0	3600
R1_4_4	7266.2	7200.12	3	3600	-	-	-	-	-	3600
R1_4_5	9184.6	9153.48	11	3600	8967.42	1	147	919	0	3600
R1_4_6	8340.4	8321.6	5	3600	7080.0	1	17	539	0	3600
R1_4_7	7599.8	7544.64	1	3600	-	-	-	-	-	3600
R1_4_8	7240.5	7161.9	1	3600	-	-	-	-	-	3600
R1_4_9	8677.5	8627.8	5	3600	8232.98	1	72	1440	0	3600
R1_6_1	21274.2	21231.91	11	3600	20773.2	1	175	441	0	3600
R1_6_10	17583.7	17344.32	3	3600	15562.7	1	14	419	0	3600
R1_6_2	18558.7	18419.8	1	3600	-	-	-	-	-	3600
R1_6_3	16874.9	16668.68	1	3600	-	-	-	-	-	3600
R1_6_4	15721.4	15538.78	1	3600	-	-	-	-	-	3600
R1_6_5	19294.9	19210.23	3	3600	18598.7	1	57	628	0	3600
R1_6_6	17763.7	17630.27	1	3600	-	-	-	-	-	3600
R1_6_7	16496.2	16300.22	1	3600	-	-	-	-	-	3600
R1_6_9	18474.1	18357.21	3	3600	17317.1	1	28	795	0	3600
R1_8_1	36345	36225.6	5	3600	35058.5	1	101	244	0	3600
R1_8_10	30918.4	30551.1	1	3600	26879.5	1	8	246	0	3600
R1_8_2	32277.6	31948.36	1	3600	-	-	-	-	-	3600
R1_8_5	33494.2	33279.06	1	3600	32087.5	1	48	641	0	3600
R1_8_6	30872.4	30460.46	1	3600	-	-	-	-	-	3600
R1_8_9	32257.3	31928.06	1	3600	29931.2	1	21	578	0	3600
R2_10_1	36881	-	-	3600	16123.8	1	21	14	0	3600
R2_2_1	3468	3468.0	1	142	3256.74	1	283	1878	0	3600
R2_2_10	2549.4	2549.4	3	1439	1322.66	1	49	136	0	3600
R2_2_2	3008.2	3008.2	1	531	941.376	1	24	72	0	3600
R2_2_3	2537.5	2537.5	1	2283	903.911	1	22	3	0	3600
R2_2_4	1928.5	1925.02	3	3600	-	-	-	-	-	3600
R2_2_5	3061.1	3061.1	1	1125	1996.68	1	92	360	0	3600
R2_2_6	2675.4	2675.4	3	2384	1023.08	1	29	74	0	3600
R2_2_7	2304.7	2298.61	1	3600	890.502	1	21	4	0	3600
R2_2_8	1842.4	1819.76	1	3600	-	-	-	-	-	3600
R2_2_9	2843.3	2843.3	1	782	1432.83	1	54	236	0	3600

Table 6 Continued

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CEIt	CESIt	CR	Cuts	Time (s)
R2_4_1	7520.7	7520.7	1	2828	5280.74	1	80	315	0	3600
R2_4_10	5645.9	5543.96	1	3600	2240.21	1	25	41	0	3600
R2_4_2	6482.8	6374.27	1	3600	1038.87	1	1	0	0	3600
R2_4_5	6567.9	6527.42	1	3600	3156.52	1	40	157	0	3600
R2_4_6	5813.5	5643.47	1	3600	-	-	-	-	-	3600
R2_4_9	6067.8	6027.42	1	3600	2614.39	1	30	114	0	3600
R2_6_1	15145.3	-	-	3600	7881.26	1	34	38	0	3600
R2_6_5	13790.2	-	-	3600	5379.07	1	20	5	0	3600
R2_6_9	12736.8	-	-	3600	2401.37	1	1	0	0	3600
R2_8_1	24969.8	-	-	3600	12277.4	1	28	32	0	3600
R2_8_5	22801.6	-	-	3600	8014.18	1	14	0	0	3600
RC1_10_1	45790.8	45302.6	1	3600	43614.7	1	35	1879	0	3600
RC1_10_5	45028.1	44404.77	1	3600	39380.4	1	13	1050	0	3600
RC1_10_6	44903.6	44284.41	1	3600	38292.0	1	9	876	0	3600
RC1_10_7	44417.1	43820.04	1	3600	34909.7	1	1	0	0	3600
RC1_10_8	43916.5	43307.23	1	3600	-	-	-	-	-	3600
RC1_10_9	43858.1	43191.82	1	3600	-	-	-	-	-	3600
RC1_2_1	3516.9	3516.9	23	2632	3441.92	2	661	2851	0	3600
RC1_2_10	2990.5	2969.39	5	3600	2819.2	1	77	1618	0	3600
RC1_2_2	3221.6	3213.54	9	3600	3051.95	1	119	1643	0	3600
RC1_2_3	3001.4	2984.21	3	3600	2742.14	1	47	800	0	3600
RC1_2_4	2845.2	2833.72	3	3600	2514.53	1	27	678	0	3600
RC1_2_5	3325.6	3319.34	9	3600	3192.13	1	261	3554	0	3600
RC1_2_6	3300.7	3300.7	3	1028	3182.13	1	290	3885	0	3600
RC1_2_7	3177.8	3154.8	5	3600	3035.01	1	169	2991	0	3600
RC1_2_8	3060	3049.85	5	3600	2926.91	1	119	2384	0	3600
RC1_2_9	3073.3	3041.67	7	3600	2908.47	1	110	2486	0	3600
RC1_4_1	8522.9	8481.66	5	3600	8258.16	1	210	2255	0	3600
RC1_4_10	7581.2	7511.6	1	3600	6654.87	1	23	1598	0	3600
RC1_4_2	7878.2	7843.85	1	3600	7089.51	1	25	875	0	3600
RC1_4_3	7516.9	7454.57	1	3600	5768.99	1	2	0	0	3600
RC1_4_4	7292.9	7206.25	1	3600	-	-	-	-	-	3600
RC1_4_5	8152.3	8101.4	1	3600	7773.46	1	91	3143	0	3600
RC1_4_6	8148	8092.64	1	3600	7758.64	1	91	3255	0	3600
RC1_4_7	7932.5	7884.28	1	3600	7443.94	1	66	2392	0	3600
RC1_4_8	7757.2	7687.88	1	3600	7073.7	1	39	2385	0	3600
RC1_4_9	7717.7	7641.23	5	3600	7004.49	1	35	2658	0	3600
RC1_6_1	16960.1	16846.82	1	3600	16278.4	1	73	1927	0	3600
RC1_6_10	15651.3	15455.46	1	3600	12949.3	1	10	959	0	3600
RC1_6_2	15890.6	15715.75	1	3600	12229.1	1	2	0	0	3600
RC1_6_3	15181.3	14922.33	1	3600	-	-	-	-	-	3600
RC1_6_4	14753.2	14405.48	1	3600	-	-	-	-	-	3600
RC1_6_5	16536.3	16377.64	1	3600	15270.3	1	36	1999	0	3600
RC1_6_6	16473.3	16315.89	1	3600	15137.4	1	36	1974	0	3600
RC1_6_7	16055.3	15929.19	3	3600	14335.9	1	24	1972	0	3600
RC1_6_8	15891.8	15689.09	1	3600	13378.6	1	13	759	0	3600
RC1_6_9	15803.5	15611.36	1	3600	13160.9	1	13	1131	0	3600
RC1_8_1	29978.9	29723.6	1	3600	28806.8	1	63	2521	0	3600
RC1_8_10	28168.5	27725.26	1	3600	22049.3	1	1	0	0	3600
RC1_8_2	28290.1	27880.5	1	3600	-	-	-	-	-	3600
RC1_8_5	29219.9	28903.94	5	3600	27127.4	1	28	1853	0	3600

Table 6 Continued

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CEIt	CESIt	CR	Cuts	Time (s)
RC1_8_6	29194.2	28795.58	5	3600	27133.9	1	29	2204	0	3600
RC1_8_7	28788.6	28400.17	3	3600	25532.1	1	17	1366	0	3600
RC1_8_8	28418.1	28007.4	1	3600	22049.3	1	1	0	0	3600
RC1_8_9	28347.1	27992.74	1	3600	-	-	-	-	-	3600
RC2_2_1	2797.4	2797.4	1	196	2191.22	1	538	2716	0	3600
RC2_2_10	1989.2	1954.78	1	3600	10.8883	1	20	515	0	3600
RC2_2_2	2481.6	2481.6	3	1808	589.51	1	26	155	0	3600
RC2_2_4	1854.8	1839.25	1	3600	-	-	-	-	-	3600
RC2_2_5	2491.4	2491.4	1	684	1221.36	1	157	968	0	3600
RC2_2_6	2495.1	2495.1	1	713	1391.57	1	245	1262	0	3600
RC2_2_7	2287.7	2284.42	5	3600	1063.54	1	155	1049	0	3600
RC2_2_8	2151.2	2151.2	1	2476	119.224	1	29	600	0	3600
RC2_2_9	2086.6	2059.93	1	3600	812.969	1	103	844	0	3600
RC2_4_1	6147.3	6141.13	1	3600	3802.14	1	162	1072	0	3600
RC2_4_2	5407.5	5328.56	1	3600	-	-	-	-	-	3600
RC2_4_5	5392.3	5369.86	1	3600	2118.76	1	50	287	0	3600
RC2_4_6	5324.6	5253.47	1	3600	2211.67	1	59	353	0	3600
RC2_4_7	4987.8	4848.73	1	3600	1099.37	1	1	0	0	3600
RC2_4_8	4693.3	4126.88	1	3600	-	-	-	-	-	3600
RC2_4_9	4510.4	2898.74	1	3600	-	-	-	-	-	3600
RC2_6_1	11966.1	-	-	3600	4797.26	1	49	344	0	3600
RC2_8_1	19201.3	-	-	3600	6242.12	1	17	211	0	3600

Appendix M: Graph Multicoloring Problem Results

In Table 7 we compare column elimination with the branch-and-price algorithm from Gualandi and Malucelli (2012) on graph multi-coloring problems. For each instance, we list the number of nodes (n) and edges (m) in the graph, and value of the maximum weighted clique (ω) used as a lower bound for preprocessing. For the branch-and-price algorithm (GM) we report the lower bound (LB), upper bound (UB), and the solution time (Time (s)), taken from the original paper. For column elimination, we report the lower bound (LB), the number of column elimination iterations (CEIt), the number of refinements (CR), and the solution time (Time (s)).

Table 7 A comparison of the branch-and-price algorithm from Gualandi and Malucelli (2012) (GM) and column elimination for solving COG instances created by Gualandi and Malucelli (2012).

Instance				GM			Column Elimination				
Name	n	m	ω	LB	UB	Time (s)	LB	UB	CEIt	CR	Time (s)
COG-10teams	3200	124480	73	-	-	3600	71	-	24	3093	3600
COG-air04	17808	2121648	377	377	377	1.8	377	377	2	0	8
COG-air05	14390	2527253	413	-	-	3600	-	-	0	0	996
COG-atlanta-ip	8124	9250	15	15	15	1844	15	-	173	4076	3600
COG-cap6000	11992	12103	14	14	14	304	14	14	2	0	1
COG-ds	15252	2057486	1	500	500	6.5	-	-	-	-	3600
COG-gesa2-o	192	144	12	12	13	3600	12	**12**	2	0	0
COG-misc07	410	2928	36	36	39	3600	36	**36**	103	378	151
COG-mkc	10394	154870	169	169	169	0.1	169	169	2	0	1
COG-mod011	192	336	12	12	13	3600	11	-	53	1716	3600
COG-mzzv11	19942	257012	101	101	101	0.1	101	101	2	0	6
COG-mzzv42z	18806	225687	91	91	91	0.1	91	91	2	0	5
COG-net12	3202	4835	17	17	17	1301	17	-	104	2600	3600
COG-nsrand-ipx	13240	69510	30	-	-	3600	30	**30**	2	0	5
COG-opt1217	1536	6528	26	-	-	3600	26	**26**	2	0	14
COG-rd-rplusc-21	904	11785	109	109	109	0.0	-	-	-	-	3600
COG-rout	560	2940	30	30	32	3600	30	**30**	2	0	0
COG-swath	12480	958000	317	-	-	3600	317	-	10	134	3600

Appendix N: Pickup-and-Delivery Problem with Time Windows Results

In Table 7 we compare column elimination with the dual ascent method from Baldacci et al. (2011a) (BBM) and VRPSolver on pickup-and-delivery problems with time windows (PDPTW). For each instance, we list the best known upper bound (UB). For BBM we report the lower bound (LB), upper bound (UB), and the solution time (Time (s)), taken from the original paper. For VRPSolver we report the lower bound (LB) and the solution time (Time (s)). For column elimination, we report the lower bound (LB), the number of column elimination iterations (CEIt) and column elimination with subgradient descent iterations (CESIt), the number of refinements (CR), and the solution time (Time (s)).

Table 9 shows the performance of column elimination for solving PDPTW instances by Li and Lim (2001) that have not yet been reported on by an exact solver.

Table 8 Comparing the dual ascent method from Baldacci et al. (2011a) (BBM), VRPSolver, and column elimination for solving a selection of PDPTW instances by Li and Lim (2001) with 200 locations.

Instance		BBM			VRPSolver		Column Elimination				
Name	UB	LB	UB	Time (s)	LB	Time (s)	LB	CEIt	CESIt	CR	Time (s)
LC1_2_1	2704.6	2704.6	2704.6	3.3	-	3600	2704.57	8	30	64	3600
LC1_2_10	2741.6	2741.6	2741.6	137.1	-	3600	2488.16	1	1425	24479	3600
LC1_2_2	2764.6	2764.6	2764.6	21.5	-	3600	2764.56	27	220	2302	3600
LC1_2_3	2772.2	2772.2	2772.2	114.9	-	3600	2694.04	1	678	7783	3600
LC1_2_4	2661.4	2395.8	2661.4	454.2	-	3600	2340.5	1	418	8118	3600
LC1_2_5	2702.0	2702.0	2702.0	4.8	-	3600	2702.05	8	40	234	10
LC1_2_6	2701.0	2701.0	2701.0	7.4	-	3600	2701.04	9	50	353	17
LC1_2_7	2701.0	2701.0	2701.0	7.7	-	3600	2701.04	8	50	355	21
LC1_2_8	2689.8	2689.8	2689.8	16.0	-	3600	2683.6	778	79708	4285	3600
LC1_2_9	2724.2	2724.2	2724.2	55.3	-	3600	2684.86	6	1942	26833	3600
LR1_2_1	4819.1	4819.1	4819.1	1.6	-	3600	4819.12	8	80	662	8
LR1_2_10	3386.3	3386.3	3386.3	1376.7	-	3600	2706.01	1	789	7763	3600
LR1_2_2	4093.1	4093.1	4093.1	20.6	-	3600	4055.38	1	400	3973	3600
LR1_2_3	3486.8	3486.8	3486.8	3690.8	-	3600	-	-	-	-	3600
LR1_2_4	2830.7	2341.8	2830.7	1809.6	-	3600	-	-	-	-	3600
LR1_2_5	4221.6	4221.6	4221.6	2.6	-	3600	4192.38	73	7758	16207	3600
LR1_2_6	3763.0	3763.0	3763.0	180.9	-	3600	3356.97	1	354	3966	3600
LR1_2_7	3112.9	2761.8	3112.9	1320.4	-	3600	1830.31	1	95	707	3600
LR1_2_8	2645.5	2150.8	2645.5	566.9	-	3600	-	-	-	-	3600
LR1_2_9	3953.5	3953.3	3953.5	15.4	-	3600	3779.64	2	2006	21428	3600
LRC1_2_1	3606.1	3606.1	3606.1	3.1	-	3600	3594.24	67	7350	5221	3600
LRC1_2_10	2837.5	2335.5	2837.5	217.4	-	3600	2181.99	1	1001	10503	3600
LRC1_2_2	3292.4	3292.4	3292.4	322.3	-	3600	3020.23	1	850	9601	3600
LRC1_2_3	3079.5	2497.8	3079.5	304.3	-	3600	-	-	-	-	3600
LRC1_2_4	2525.8	1981.0	2525.8	188.2	-	3600	-	-	-	-	3600
LRC1_2_5	3715.8	3715.8	3715.8	42.1	-	3600	3280.25	1	2845	41795	3600
LRC1_2_6	3360.9	3360.9	3360.9	7.0	-	3600	3205.94	2	2907	35316	3600
LRC1_2_7	3317.7	3317.7	3317.7	408.2	-	3600	2795.52	1	2310	30413	3600
LRC1_2_8	3086.5	3086.5	3086.5	1562.7	-	3600	2386.28	2	1636	20021	3600
LRC1_2_9	3053.8	3053.8	3053.8	1757.2	-	3600	2375.2	4	1479	19662	3600
LC1_10_1	42488.66	42488.7	42488.7	79.5	-	3600	42488.7	8	70	951	149
LC1_10_5	42477.4	42477.4	42477.4	118.7	-	3600	42477.4	8	210	4843	2232
LR1_10_1	56744.91	56744.9	56744.9	233.1	-	3600	55200.4	1	639	29085	3600
LR1_10_5	59053.68	52536.3	52901.3	4068.8	-	3600	43578.9	1	163	7065	3600
LRC1_10_1	49111.78	48398.8	48666.5	2533.3	-	3600	38724.2	1	186	9161	3600
LRC1_10_5	50323.04	38177.8	49287.1	1650.3	-	3600	2112.59	1	22	3865	3600

Table 9 The performance of column elimination for solving PDPTW instances by Li and Lim (2001) that have not yet been reported on by an exact solver.

Instance		Column Elimination				
Name	UB	LB	CEIt	CESIt	CR	Time (s)
LC1_10_6	42838.39	42512.1	1	269	11426	3600
LC1_10_7	42854.99	42482.7	1	360	14140	3600
LC1_10_8	42949.56	36753.0	1	99	5731	3600
LC1_10_9	43564.01	10545.2	1	3	296	3600
LC1_4_1	7152.06	**7152.06**	8	30	152	15
LC1_4_10	7850.22	6048.13	1	265	8047	3600
LC1_4_2	8007.79	6849.74	1	131	2031	3600
LC1_4_3	8678.23	5839.12	1	81	1394	3600
LC1_4_4	6451.68	4955.73	1	52	798	3600
LC1_4_5	7150.0	**7150.0**	8	50	435	54
LC1_4_6	7154.02	**7154.02**	16	80	1218	1349
LC1_4_7	7149.43	**7149.43**	11	90	1583	712
LC1_4_8	8305.42	7077.22	6	1171	24073	3600
LC1_4_9	7451.2	6242.41	1	230	7154	3600
LC1_6_1	14095.64	**14095.6**	9	40	328	71
LC1_6_2	15048.16	12851.8	1	82	2113	3600
LC1_6_3	14874.68	10270.8	1	44	856	3600
LC1_6_5	14086.3	**14086.3**	8	90	1307	358
LC1_6_6	14090.79	**14090.8**	8	150	2969	735
LC1_6_7	14083.76	**14083.8**	9	120	2791	2310
LC1_6_8	14880.7	12983.3	1	114	4320	3600
LC1_6_9	15207.58	11524.3	1	79	2442	3600
LC1_8_1	25184.38	**25184.4**	8	50	591	92
LC1_8_2	29735.85	5436.04	1	2	186	3600
LC1_8_5	25211.22	**25211.2**	8	100	2398	828
LC1_8_6	25164.25	**25164.3**	12	210	4882	3009
LC1_8_7	25158.38	25158.3	6	190	6585	3600
LC1_8_8	26688.17	21526.2	1	59	2391	3600
LC1_8_9	26975.61	17537.6	1	40	828	3600
LC2_10_1	16879.24	16834.4	2	90	1269	3600
LC2_10_10	17043.64	13641.0	1	72	1388	3600
LC2_10_5	17137.53	15746.7	1	232	6644	3600
LC2_10_6	19387.75	15088.3	1	142	3760	3600
LC2_10_7	18389.37	14776.0	1	96	2005	3600
LC2_10_8	17015.03	14450.1	1	102	2355	3600
LC2_10_9	18225.3	10031.1	1	22	40	3600
LC2_2_1	1931.44	**1931.44**	16	100	478	86
LC2_2_10	1817.45	1706.59	1	796	5085	3600
LC2_2_2	1881.4	1854.18	2	669	3844	3600
LC2_2_3	1844.33	1701.93	1	282	2046	3600
LC2_2_4	1767.12	1517.13	1	143	705	3600
LC2_2_5	1891.21	1877.31	10	1885	13262	3600
LC2_2_6	1857.78	1804.46	1	1194	8664	3600
LC2_2_7	1850.13	1806.89	2	907	6338	3600
LC2_2_8	1824.34	1754.23	1	1009	7019	3600
LC2_2_9	1854.21	1754.86	1	817	5696	3600
LC2_4_1	4116.33	**4116.33**	10	150	1181	563
LC2_4_10	3828.44	3412.01	1	182	1865	3600

Table 9 Continued

Instance		Column Elimination				
Name	UB	LB	CEIt	CESIt	CR	Time (s)
LC2_4_2	4144.29	3916.04	1	245	3061	3600
LC2_4_3	4401.08	3128.95	1	81	533	3600
LC2_4_4	3743.95	1793.21	1	27	43	3600
LC2_4_5	4030.63	3884.4	1	516	6415	3600
LC2_4_6	3900.29	3687.84	1	255	3015	3600
LC2_4_7	3962.51	3694.87	1	284	3597	3600
LC2_4_8	3844.45	3586.93	1	284	3394	3600
LC2_4_9	4188.93	3527.43	1	181	1814	3600
LC2_6_1	7977.98	7876.53	1	556	5794	3600
LC2_6_10	7946.6	6371.09	1	103	1450	3600
LC2_6_2	9900.48	7331.11	1	159	2502	3600
LC2_6_3	8512.94	5518.71	1	45	217	3600
LC2_6_5	9051.53	7344.22	1	446	8306	3600
LC2_6_6	8775.55	7037.1	1	242	4162	3600
LC2_6_7	9376.58	6875.76	1	117	1800	3600
LC2_6_8	7579.63	6767.95	1	167	2550	3600
LC2_6_9	8714.22	6763.4	1	135	1968	3600
LC2_8_1	11687.06	11649.6	9	120	1372	3600
LC2_8_10	12226.42	9442.24	1	73	965	3600
LC2_8_2	13713.13	9562.29	1	45	228	3600
LC2_8_5	12298.33	11068.8	1	264	5907	3600
LC2_8_6	12645.71	10566.1	1	179	3769	3600
LC2_8_7	14041.47	10317.8	1	100	1590	3600
LC2_8_8	12924.1	10045.1	1	105	1653	3600
LC2_8_9	11629.41	9919.23	1	87	1354	3600
LR1_10_9	54582.25	35187.2	1	79	2017	3600
LR1_4_1	10639.75	**10639.8**	9	530	3906	437
LR1_4_10	8192.65	126.225	1	39	2340	3600
LR1_4_5	11374.06	9381.73	4	1274	21471	3600
LR1_4_9	9859.47	7648.39	1	429	8360	3600
LR1_6_1	22821.65	22452.5	7	1194	17635	3600
LR1_6_10	19298.25	10244.8	1	92	1551	3600
LR1_6_5	23623.52	18469.2	1	370	11561	3600
LR1_6_9	21835.87	14816.1	1	158	3374	3600
LR1_8_1	39291.32	38716.1	1	949	25746	3600
LR1_8_5	39809.68	30320.6	1	210	8448	3600
LR1_8_9	38499.0	24194.6	1	113	2555	3600
LR2_2_1	4073.1	3346.21	1	296	1633	3600
LR2_2_10	3254.83	1279.46	1	60	44	3600
LR2_2_2	3796.0	908.641	1	40	19	3600
LR2_2_3	3098.36	197.901	1	1	0	3600
LR2_2_5	3438.39	1922.72	1	107	261	3600
LR2_2_6	4457.95	1074.38	1	47	10	3600
LR2_2_9	3922.11	1396.92	1	70	134	3600
LR2_4_1	9726.88	4821.87	1	82	159	3600
LR2_4_10	7596.62	1937.65	1	32	0	3600
LR2_4_5	9894.46	2861.37	1	53	55	3600
LR2_4_9	7926.07	2605.59	1	44	33	3600
LR2_6_1	21759.33	7051.15	1	47	27	3600
LR2_6_5	22625.89	1248.95	1	1	0	3600

Table 9 Continued

Instance		Column Elimination				
Name	UB	LB	CEIt	CESIt	CR	Time (s)
LR2_6_9	18259.2	1260.05	1	2	0	3600
LR2_8_1	41873.12	10640.0	1	38	1	3600
LRC1_10_6	45115.22	28409.1	1	128	6034	3600
LRC1_4_1	9124.52	8786.89	9	2219	27012	3600
LRC1_4_5	8847.4	7362.81	1	1208	32476	3600
LRC1_4_6	8394.47	7055.52	1	1315	33591	3600
LRC1_4_7	8037.87	5980.69	1	787	18678	3600
LRC1_4_8	7930.15	5244.7	1	540	11088	3600
LRC1_4_9	8004.24	5290.96	1	678	14613	3600
LRC1_6_1	18288.9	17196.9	1	993	27370	3600
LRC1_6_5	17463.94	13599.9	1	470	15706	3600
LRC1_6_6	19025.36	12819.5	1	473	15816	3600
LRC1_6_7	15914.85	11060.8	1	277	8521	3600
LRC1_8_1	32252.28	29266.0	1	477	21035	3600
LRC1_8_5	31169.16	20067.0	1	180	7759	3600
LRC1_8_6	28961.66	3690.2	1	53	6481	3600
LRC2_2_1	3595.18	2208.29	1	555	2737	3600
LRC2_2_10	2437.88	688.411	1	74	388	3600
LRC2_2_2	3158.25	568.591	1	35	74	3600
LRC2_2_5	2776.93	1378.96	1	206	806	3600
LRC2_2_6	2707.96	1290.34	1	243	1037	3600
LRC2_2_7	3010.68	1038.16	1	133	649	3600
LRC2_2_8	2399.89	878.886	1	105	460	3600
LRC2_2_9	2208.49	932.522	1	115	481	3600
LRC2_4_1	9738.95	3925.03	1	189	1034	3600
LRC2_4_10	5395.71	591.091	1	1	0	3600
LRC2_4_5	7309.54	2224.78	1	76	255	3600
LRC2_4_6	6337.08	2343.81	1	88	303	3600
LRC2_4_7	6292.23	1618.1	1	36	81	3600
LRC2_4_9	6272.92	1585.12	1	39	90	3600
LRC2_6_1	14578.92	4764.83	1	73	339	3600
LRC2_8_1	23074.22	2250.61	1	1	0	3600

Appendix O: Sequential Ordering Problem Results

In Table 10 we compare column elimination with the tree-search solver from (Libralesso et al. 2020) (CATS) on sequential ordering problem (SOP) instances from TSPLIB. For each instance, we list the best known upper bound (UB). For CATS we report the lower bound (LB) and the solution time (Time (s)), taken from the original paper. For column elimination, we report the lower bound (LB), the number of column elimination iterations (CEIt) and column elimination with subgradient descent iterations (CESIt), the number of refinements (CR), and the solution time (Time (s)).

Table 10 Comparing the performance of the tree-search solver from (Libralesso et al. 2020) (CATS) and column elimination for solving SOP instances from TSPLIB.

Instance		CATS		Column Elimination				
Name	UB	LB	Time (s)	LB	CEIt	CESIt	CR	Time (s)
ESC07	2125	2125	0	2125.0	24	1	32	0
ESC11	2075	2075	0	2075.0	12	1	26	0
ESC12	1675	1675	0	1675.0	180	1	1714	3
ESC25	1681	818	3600	1681.0	320	1	6177	274
ESC47	1288	315	3600	998.1	126	60	2526	3600
ESC63	62	4	3600	54.9	198	20673	20476	3600
ESC78	18230	850	3600	10821.4	236	30070	29835	3600
br17.10	55	55	59	55.0	1680	1	12355	1145
br17.12	55	55	25	55.0	1543	1	12864	1173
ft53.1	7531	1036	3600	6825.8	96	13800	13705	3600
ft53.2	8026	1136	3600	6376.1	148	17088	16941	3600
ft53.3	10262	2119	3600	7086.2	535	57807	57273	3600
ft53.4	14425	14425	241	9246.8	838	88421	87584	3600
ft70.1	39313	3094	3600	38345.7	87	9216	9130	3600
ft70.2	40419	2877	3600	38543.8	110	11592	11483	3600
ft70.3	42535	4490	3600	39380.9	338	35000	34663	3600
ft70.4	53530	20269	3600	43999.1	480	49384	48905	3600
kro124p.1	39420	2467	3600	34870.7	30	4380	4351	3600
kro124p.2	41336	3116	3600	34971.7	42	5756	5715	3600
kro124p.3	49499	4065	3600	36040.5	174	19699	19526	3600
kro124p.4	76103	7150	3600	42348.2	229	26183	25955	3600
p43.1	28140	215	3600	4545.2	79	40511	40433	3600
p43.2	28480	295	3600	4649.4	91	47493	47403	3600
p43.3	28835	585	3600	4750.2	207	64027	63821	3600
p43.4	83005	83005	183	6541.4	901	123460	122560	3600
prob.100	1163	149	3600	793.1	1	4160	4160	3600
prob.42	243	82	3600	225.8	131	70	2219	3600
rbg048a	351	63	3600	322.3	201	23482	23282	3600
rbg050c	467	70	3600	428.0	386	43948	43563	3600
rbg109a	1038	117	3600	927.9	107	17006	16900	3600
rbg150a	1750	123	3600	1132.0	1	7376	7376	3600
rbg174a	2033	471	3600	1251.6	1	5234	5233	3600
rbg253a	2950	140	3600	1267.3	1	1292	1292	3600
rbg323a	3140	74	3600	1262.1	1	726	726	3600
rbg341a	2568	67	3600	67.0	1	637	636	3600
rbg358a	2545	101	3600	77.4	1	451	451	3600
rbg378a	2816	66	3600	51.6	1	446	446	3600
ry48p.1	15805	2856	3600	14220.1	135	14401	14267	3600
ry48p.2	16666	3779	3600	14383.6	184	19446	19263	3600
ry48p.3	19894	5268	3600	15094.8	487	50826	50340	3600
ry48p.4	31446	31446	57	22176.1	1148	120079	118932	3600

Appendix P: Graph Coloring Problem Results

In Table 10 we compare column elimination with the branch-and-price algorithm by (Held et al. 2011) (Held et al.) on graph coloring instances from the DIMACS benchmark set. For each instance, we list the number of nodes (n) and edges (m), and the best known solution (BKS). For both methods we report the lower bound (LB) and the solution time (Time (s)). Both methods are run on the same server.

Table 11 A comparison of the branch-and-price algorithm by (Held et al. 2011) and column elimination on graph coloring problem instances from the DIMACS benchmark set.

Instance				Held et al.		Column Elimination	
Name	n	m	BKS	LB	Time (s)	LB	Time (s)
1-FullIns_3	30	100	4	4	0	4	0
1-FullIns_4	93	593	5	4	3600	5	5
1-FullIns_5	282	3247	6	5	3600	6	3469
1-Insertions_4	67	232	5	4	3600	3	3600
1-Insertions_5	202	1227	6	5	3600	3	3600
1-Insertions_6	607	6337	7	6	3600	3	3600
2-FullIns_3	52	201	5	5	0	5	0
2-FullIns_4	212	1621	6	5	3600	6	3
2-FullIns_5	852	12201	7	6	3600	5	3600
2-Insertions_3	37	72	4	4	206	3	3600
2-Insertions_4	149	541	5	4	3600	3	3600
2-Insertions_5	597	3936	6	5	3600	3	3600
3-FullIns_3	80	346	6	6	0	6	0
3-FullIns_4	405	3524	7	6	3600	7	36
3-FullIns_5	2030	33751	8	6	3600	5	3600
3-Insertions_3	56	110	4	3	3600	3	3600
3-Insertions_4	281	1046	5	4	3600	3	3600
3-Insertions_5	1406	9695	6	3	3600	2	3600
4-FullIns_3	114	541	7	7	0	7	0
4-FullIns_4	690	6650	8	7	3600	8	65
4-FullIns_5	4146	77305	9	7	3600	6	3600
4-Insertions_3	79	156	4	3	3600	3	3600
4-Insertions_4	475	1795	5	4	3600	3	3600
5-FullIns_3	154	792	8	8	0	8	0
5-FullIns_4	1085	11395	9	8	3600	9	277
C2000.5	2000	999836	145	1	3600	12	3600
C2000.9	2000	1799532	400	1	3600	103	3600
C4000.5	4000	4000268	259	1	3600	1	3600
DSJC1000.1	1000	49629	20	1	3600	5	3600
DSJC1000.5	1000	249826	82	1	3600	14	3600
DSJC1000.9	1000	449449	222	217	3600	1	3600
DSJC125.1	125	736	5	5	1	5	31
DSJC125.5	125	3891	17	17	129	13	3600
DSJC125.9	125	6961	44	44	1	43	3600
DSJC250.1	250	3218	8	7	3600	5	3600
DSJC250.5	250	15668	28	27	3600	14	3600
DSJC250.9	250	27897	72	72	1757	71	3600

Table 11 Continued

Instance				Held et al.		Column Elimination	
Name	n	m	BKS	LB	Time (s)	LB	Time (s)
DSJC500.1	500	12458	12	1	3600	5	3600
DSJC500.5	500	62624	47	44	3600	14	3600
DSJC500.9	500	224874	126	125	3600	118	3600
DSJR500.1	500	3555	12	12	0	12	32
DSJR500.1c	500	121275	85	85	9	85	66
DSJR500.5	500	58862	122	122	29	119	3600
abb313GPIA	1557	65390	9	8	3600	7	3600
anna	138	986	11	11	0	11	0
ash331GPIA	662	4185	4	4	1	3	3600
ash608GPIA	1216	7844	4	4	856	2	3600
ash958GPIA	1916	12506	4	1	3600	3	3600
david	87	812	11	11	0	11	0
flat1000_50_0	1000	245000	50	50	1658	13	3600
flat1000_60_0	1000	245830	60	1	3600	14	3600
flat1000_76_0	1000	246708	81	1	3600	14	3600
flat300_20_0	300	21375	20	20	4	13	3600
flat300_26_0	300	21633	26	26	14	13	3600
flat300_28_0	300	21695	28	28	9	13	3600
fpsol2.i.1	496	11654	65	65	0	65	6
fpsol2.i.2	451	8691	30	30	0	30	37
fpsol2.i.3	425	8688	30	30	0	30	37
games120	120	1276	9	9	0	9	34
homer	561	3258	13	13	0	13	1
huck	74	602	11	11	0	11	0
inithx.i.1	864	18707	54	54	1	54	46
inithx.i.2	645	13979	31	31	0	31	64
inithx.i.3	621	13969	31	31	0	31	65
jean	80	508	10	10	0	10	0
latin_square_10	900	307350	98	91	3600	90	3600
le450_15a	450	8168	15	15	8	13	3600
le450_15b	450	8169	15	15	9	15	31
le450_15c	450	16680	15	15	166	11	3600
le450_15d	450	16750	15	1	3600	11	3600
le450_25a	450	8260	25	25	1	20	3600
le450_25b	450	8263	25	25	1	25	34
le450_25c	450	17343	25	25	13	22	3600
le450_25d	450	17425	25	25	12	19	3600
le450_5a	450	5714	5	5	677	5	32
le450_5b	450	5734	5	5	706	5	32
le450_5c	450	9803	5	5	131	5	37
le450_5d	450	9757	5	5	65	5	30
miles1000	128	6432	42	42	0	42	0
miles1500	128	10396	73	73	0	73	0
miles250	128	774	8	8	0	8	0
miles500	128	2340	20	20	0	20	0
miles750	128	4226	31	31	0	31	0
mug100_1	100	166	4	4	0	1	3600
mug100_25	100	166	4	4	0	1	3600
mug88_1	88	146	4	4	0	3	3600
mug88_25	88	146	4	4	0	1	3600

Table 11 Continued

Instance				Held et al.		Column Elimination	
Name	n	m	BKS	LB	Time (s)	LB	Time (s)
mulsol.i.1	197	3925	49	49	0	49	0
mulsol.i.2	188	3885	31	31	0	31	5
mulsol.i.3	184	3916	31	31	0	31	5
mulsol.i.4	185	3946	31	31	0	31	8
mulsol.i.5	186	3973	31	31	0	31	13
myciel3	11	20	4	4	0	4	0
myciel4	23	71	5	5	4	5	2
myciel5	47	236	6	5	3600	4	3600
myciel6	95	755	7	6	3600	4	3600
myciel7	191	2360	8	7	3600	3	3600
qg.order100	10000	990000	100	1	3600	1	3600
qg.order30	900	26100	30	30	78	30	49
qg.order40	1600	62400	40	40	480	40	90
qg.order60	3600	212400	60	1	3600	60	2547
queen10_10	100	2940	11	11	109	10	3600
queen11_11	121	3960	11	11	0	11	30
queen12_12	144	5192	12	12	0	12	30
queen13_13	169	6656	13	13	1	13	33
queen14_14	196	8372	14	14	1	14	34
queen15_15	225	10360	15	15	2	15	30
queen16_16	256	12640	16	16	4	16	30
queen5_5	25	320	5	5	0	5	0
queen6_6	36	580	7	7	0	7	0
queen7_7	49	952	7	7	0	7	0
queen8_12	96	2736	12	12	0	12	31
queen8_8	64	1456	9	9	0	9	32
queen9_9	81	2112	10	10	3	9	3600
r1000.1	1000	14378	20	20	0	20	44
r1000.1c	1000	485090	97	96	3600	85	3600
r1000.5	1000	238267	234	234	380	227	3600
r125.1	125	209	5	5	0	5	0
r125.1c	125	7501	46	46	0	46	0
r125.5	125	3838	36	36	0	36	57
r250.1	250	867	8	8	0	8	0
r250.1c	250	30227	64	64	0	64	3
r250.5	250	14849	65	65	2	64	3600
school1	385	19095	14	14	53	1	3600
school1_nsh	352	14612	14	14	18	1	3600
wap01a	2368	110871	41	1	3600	39	3600
wap02a	2464	111742	41	1	3600	38	3600
wap03a	4730	286722	44	1	3600	38	3600
wap04a	5231	294902	42	1	3600	1	3600
wap05a	905	43081	50	50	4	40	3600
wap06a	947	43571	40	40	106	35	3600
wap07a	1809	103368	41	1	3600	38	3600
wap08a	1870	104176	41	1	3600	34	3600
will199GPIA	701	7065	7	7	1	6	3600
zeroin.i.1	211	4100	49	49	0	49	0
zeroin.i.2	211	3541	30	30	0	30	0
zeroin.i.3	206	3540	30	30	0	30	0