2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Cooperative Approach between Metaheuristic and Branch-and-Price for the Team Orienteering Problem with Time Windows

Liangjun Ke
State Key Laboratory for
Manufacturing Systems Engineering
Xian Jiaotong University
Xi'an, China 710049
Email: keljxjtu@xjtu.edu.cn

Huimin Guo
Systems Engineering Institute
Xian Jiaotong University
Xi'an, China 710049
Email: ghmlpx@stu.xjtu.edu.cn

Qingfu Zhang
Department of Computer Science
City University of Hong Kong
Hong Kong
Email: qingfu.zhang@cityu.edu.hk

***Abstract*—The team orienteering problem with time windows (TOPTW) is a well studied routing problem. In this paper, a cooperative algorithm is proposed. It collaborates metaheuristic and branch-and-price. A restricted master problem and subproblem are defined. It uses a heuristic to obtain an integral solution for the restricted master problem and a metaheuristic to generate new columns for the subproblem. Experimental study shows that this algorithm can find new better solutions for several instances in short time, which supports the effectiveness of the cooperative mechanism between metaheuristic and branch-and-price.**

I. Introduction

Although the optimization arsenal has become increasingly rich, solving to optimality is still a big challenge. Exact algorithms such as branch-and-price (B-P) can, at least in theory, find an optimal solution if it exists. However, for a lot of optimization problems, especially, those so-called NP-hard problems, the exact algorithms are not applicable in practice due to the limitation of computation resources (e.g., CPU time and memory).

The team orienteering problem with time windows (TOPTW) [1] is such a NP-hard problem, extended from orienteering problem [2]. In the TOPTW, a fixed number of vehicles are required to serve a set of nodes, each of which has a reward and a time window. Once a node is served by a vehicle within its time window, the reward of this node will be received. The objective is to maximize the total received reward while the travel time of each path must be not more than a time limit. Even for the TOP, a special case of TOPTW without time window constraints, B-P only can solve very limited size instances [3]. A practicable approach is to resort to approximate algorithms, which often can find a satisfactory solution in a reasonable amount of running time at the cost of the optimality.

So far, several approximate algorithms have been developed [4]. An ant colony system algorithm was proposed in [1] and extended in [5], [6]. In [7], a variable neighborhood search (VNS) was proposed for the TOPTW. An iterated local search was proposed in [8]. In [9], iterated local search was combined with a greedy randomized adaptive search procedure for the multi-constraint team orienteering problem with multiple time windows, which can be viewed as a variant of the TOPTW, and the experimental results on the TOPTW demonstrate that the hybrid algorithm is effective for the TOPTW. A hybrid metaheuristic which combines the greedy randomized adaptive search procedure with the evolutionary local search approach was developed in [10]. Two simulated annealing algorithms were proposed in [11]. A VNS algorithm which explores granular neighborhoods (GVNS) based on linear programming was introduced in [12]. An iterative framework incorporating three components was developed in [13]. It uses a local search procedure and a simulated annealing procedure to generate a set of paths, and the third procedure to recombine the paths to find high quality solutions. In [14], a bee colony algorithm was proposed. A cluster-based algorithm was introduced in [15].

In this paper, a new approach is proposed which cooperates metaheuristic with B-P. B-P is a popular exact algorithm which combines column generation and branch-and-bound [16]. B-P starts from a subset of columns and formulates a restricted master problem (RMP) and a subproblem using the dual information obtained by solving the RMP. It gradually adds new columns by solving a subproblem and updates the RMP. These two steps stop repeating once no new column can be generated. If the resulting solution is integral, the algorithm stops. Otherwise, the branch and bound is invoked. In the exact version, an exact algorithm, (e.g. dynamic programming) is used for generating columns, which makes B-P very slow. Since metaheuristics can find a promising solution for many complex optimization problems, we use a metaheuristic, instead of an exact algorithm, for column generation. In other words, the metaheuristic is used to solve the subproblem. Once metaheuristic fails to generate new columns, a heuristic is used to generate an integral solution and update the bound. We observe that some efforts have been made to combine exact algorithms and approximate algorithms. A good survey can be found in [17]. To the best of our knowledge, no similar algorithm has been developed and much less applied on the TOPTW.

The remainder of the paper is structured as follows. Section II describes the problem model. Section III presents the cooperative algorithm. Section IV presents the experimental results. Finally, the main results are concluded in section V.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

II. Problem formulation

The TOPTW is defined on an undirected complete graph $G = (V, E)$, where $V = \{0, 1, \cdots, n+1\}$ is the set of nodes and $E = \{(i, j) \,|i, j \in V\}$ is the set of edges. Each edge $(i, j) \in E$ is associated with a travel time c_{ij}. Each node i has a reward r_i and a time window $[o_i, c_i]$. A vehicle must arrive at node i before c_i. If it arrives before o_i, i.e., the time window does not open, it should wait until o_i to serve node i. A path is said to be feasible if it starts at node 0 and finishes at node $n+1$, and its travel time cannot exceed a time limit $T_{\max}$, moreover, each node of the path is visited in its time window. Let Ω be the set of all feasible paths.

Let $R(r)$ be the total received reward of a path r. Binary variable $a_{ir} = 1$ if path $r \in \Omega$ visits node i $(1 \leq i \leq n)$, otherwise $a_{ir} = 0$. Binary variable $x_r = 1$ if path $r \in \Omega$ is traveled, otherwise $x_r = 0$. The model of the TOPTW can be described as follows:

$$\text{maximize} \quad \sum_{r \in \Omega} R(r) x_r \tag{1}$$

$$\text{subject to} \quad \sum_{r \in \Omega} a_{ir} x_r \leq 1, \;\; 1 \leq i \leq n \tag{2}$$

$$\sum_{r \in \Omega} x_r \leq m \tag{3}$$

$$x_r \in \{0, 1\}, \forall r \in \Omega \tag{4}$$

In the TOPTW, the objective value of solution $x = (x_r)_{r \in \Omega}$ is defined as $f(x) = \sum_{r \in \Omega} R(r) x_r$. The objective function (1) maximizes the total received reward. The constraints (2) guarantee that each node only can be visited at most once, and (3) ensure that a feasible solution consists of at most m paths, and (4) are the integrality constraints.

III. The cooperative algorithm

This section first describes the restricted master problem and subproblem, and then introduces the cooperative algorithm.

A. The restricted master problem and subproblem

The restricted master problem is stated as follows:

$$\text{maximize} \quad \sum_{r \in \Omega'} R(r) x_r \tag{5}$$

$$\text{subject to} \quad \sum_{r \in \Omega'} a_{ir} x_r \leq 1, \;\; 1 \leq i \leq n \tag{6}$$

$$\sum_{r \in \Omega'} x_r \leq m \tag{7}$$

$$x_r \in \{0, 1\}, \forall r \in \Omega' \tag{8}$$

where $\Omega' \in \Omega$ is a subset of columns. In our algorithm, we solve the linear relaxation of the restricted master problem in which constraints (8) is replaced by the following constraints:

$$0 \leq x_r \leq 1, \forall r \in \Omega' \tag{9}$$

Let λ_i $(1 \leq i \leq n)$ denotes the nonnegative dual variable corresponding to constraints $\sum_{r \in \Omega'} a_{ir} x_r \leq 1 \, (1 \leq i \leq n)$ for node i and λ_0 the nonnegative dual variable corresponding to constraint $\sum_{r \in \Omega'} x_r \leq m$. The subproblem aims to find a feasible path r with a maximum positive reduced cost. The reduced cost satisfies the following conditiofischetti1998solvingn:

$$g(r) = R(r) - \sum_{i \in J} a_{ir} \lambda_i - \lambda_0 > 0 \tag{10}$$

where $g(r)$ is the fitness of path r, $J = \{1, \cdots, n\}$.

Algorithm 1: The cooperative Algorithm

```
input : a TOPTW instance
output: the best so far solution s*
initialize Restricted Master problem RMP
initialize the set of columns Ω' with simple set of paths
  and generate the root of the list of nodes, denoted as
  NodeQueue
initialize the best so far solution s*
push root into NodeQueue
while NodeQueue is not empty do
    node ← the top node of NodeQueue
    pop up the top node of NodeQueue
    [s, Ω'] ← ColumnGeneration(node, Ω') /*see
    Algorithm 2*/
    if each component of s is integral then
        if s is better than s* then
            s* ← s
        end
    else
        generate an integral solution and update the
        bound by Algorithm 4
        perform the branching and add nodes into
        NodeQueue
    end
    update NodeQueue
end
```

Algorithm 2: Column generation procedure

```
input : node, Ω'
output: Ω'
do
    determine RMP
    solve linear relaxation of RMP
    generate the dual values λ_i(0 ≤ i ≤ n) of RMP
    obtain a set of columns with positive reduced cost,
    denoted as A, by Algorithm 3
    add A into Ω'
while A is not empty
```

B. Outline of the cooperative algorithm

Our algorithm initializes n columns, each of which only visits one node, and forms a root node. For the node, two steps are repeated. It first obtains the dual values by solving the linear relaxation of the restricted master problem. After that, the subproblem is renewed by these dual values, and a metaheuristic is called with the aim of finding new columns with positive reduced costs. Once no new path can be obtained, the node is set to be inactive. If the optimal solution of

Algorithm 3: The VNS procedure for generating new columns

input : a path x
output: a new path x
$k \leftarrow 1, it \leftarrow 0, flag = 0$
while $it < NoImp$ **do**
  $x' \leftarrow Shake(x, k) / * x' \in N_k(x) * /$
  $x'' \leftarrow Localsearch(x')$
  if $g(x'') > g(x)$ **then**
    $x \leftarrow x''$
    $k \leftarrow 1$
    $it \leftarrow 0$
    $flag \leftarrow 1$
  else
    $k \leftarrow \max(k+1, k_{max}+1)$
    if $k > k_{max}$ **then**
      $k \leftarrow 1$
    end
    $it \leftarrow it + 1$
  end
end
if $flag = 1$ **then**
  add x into Ω'
end

Algorithm 4: The heuristic for generating an integral solution and updating the bound

input : $\Lambda = \{r \in \Omega' | x_r > 0\}, bound$
output: an integral solution $s*, bound$
$s* = \emptyset$
set the objective value of $s*$ as 0
for *each* $r \in \Lambda$ **do**
  set r as the first path of a solution s
  $it \leftarrow 1$
  while $it < m$ **do**
    $\overline{\Lambda} \leftarrow \Lambda$
    for *each* $r' \in \overline{\Lambda}$ **do**
      remove from r' the same nodes in s and replace r' by the resulting path
    end
    find the path $r_{best} \in \overline{\Lambda}$ with the largest total reward
    add r_{best} into s
    $it \leftarrow it + 1$
  end
  if $f(s) > f(s*)$ **then**
    $s* \leftarrow s$
  end
end
if $f(s*) > bound$ **then**
  $bound \leftarrow f(s*)$
end

the linear relaxation of the restricted master problem is not integral, a branching procedure is called, and new nodes are formed and added into the list of nodes, denoted as $NodeQueue$. The pseudo-code of the cooperative algorithm is given in Algorithm 1.

The initialization step is presented in the lines (1-4). At line 8, the RMP is solved by the column generation procedure (see Algorithm 2). After that, if the final solution is integral, the best so far solution is updated, otherwise, a heuristic is carried out in line 14 for generating an integral solution. Then a branching procedure is adopted once the solution is fractional. The algorithm stops when $NodeQueue$ is empty.

The column generation procedure is presented in Algorithm 2. Note that RMP depends on the set of columns Ω', and each column corresponds to a feasible path. The dual information is updated in line 4. Line 5 generates new columns by Algorithm 3. The algorithm stops when no column with positive reduced cost can be generated.

In line 15, the child nodes are generated by branching. If $\sum_{r \in \Omega}(a_{ir} r)$ is fractional, two new branches are generated. Otherwise, the arc strategy is used. In this case, if the flow of arc (k, l) is fractional, three new branches are generated. More details are available in [3]. In the cooperative algorithm, if the size of $NodeQueue$ is less than 30, all new nodes are added. Otherwise, only one node selected randomly is added into $NodeQueue$ and the others are added into an auxiliary list $AuxQueue$. Finally, in line 17, those nodes in $NodeQueue$ whose upper bound is smaller than the integral bound $bound$ are discarded. If $NodeQueue$ is empty, the nodes in $AuxQueue$ are added into $NodeQueue$.

C. The metaheuristic for the subproblem

In our implementation, a variable neighborhood search (VNS) is adopted to generate new columns since VNS is simple and effective [18]. It finds a local optimum in one neighborhood and uses a shaking move to get out of the local basin and then searches in a new neighborhood by local search. For a path r, its neighborhood $N_k (k = 1, \cdots, k_{max})$ is defined by removing k nodes from it and inserting new nodes as many as possible, where k_{max} is a parameter. Let $\Lambda = \{r \in \Omega' | x_r > 0\}$. VNS starts from each path in Λ and iteratively improves it with the attempt of finding a better path. The shaking move randomly selects a solution from the neighborhood N_k of the incumbent solution. The local search determines the first-best solution in the neighborhood N_1 of the incumbent solution and updates it. Local search stops repeating until no improved incumbent solution can be found. The detail of VNS is given in Algorithm 3.

D. Generating an integral solution for the restricted master problem

We use a heuristic to generate an integral solution. At first, we obtain the optimal solution of the linear relaxation of the restricted master problem. We try to use each path in Λ as the first path of the integral solution. After that, for each path in Λ, those nodes included in the integral solution are removed. From the resulting paths, the path with the maximum reward is selected as the next path of the integral solution. This procedure is repeated until no path can be added into the solution. The final integral solution is the solution with the largest total reward. The detail of the heuristic is given in

TABLE I. THE RESULTS ON THE TOTAL REWARD OBTAINED FOR THE TEN INSTANCES WHEN THE TIME LIMIT IS 120S

Instance			$Best\ known$	The cooperative algorithm			GVNS		
$name$	n	m		Worst	Average	Best	Worst	Average	Best
pr01	48	4	657	635	643.4	651	654	655.8	*657*
pr02	96	4	1079	1056	1061.4	1064	1055	1064.2	1070
pr03	144	4	1232	1234	1239.2	**1243**	1202	1209.6	1215
pr04	192	4	1585	1547	1560.2	1571	1522	1525.8	1531
pr05	240	4	1838	1768	1805.6	1825	1785	1811.4	1827
pr06	288	4	1860	1837	1852.2	**1875**	1826	1840.4	1855
pr07	72	4	876	876	876.0	*876*	855	862.6	870
pr08	144	4	1382	1337	1358.8	**1384**	1355	1358.2	1361
pr09	216	4	1619	1606	1608.8	1610	1589	1595.4	1607
pr10	288	4	1943	1862	1874.4	1891	1894	1903.6	1916

Boldface indicates a new best result is obtained
Italic indicates the same best-so-far results is obtained

TABLE II. THE RUNNING TIME (IN SECONDS) CONSUMED BY THE COMPARED ALGORITHMS FOR THE TEN INSTANCES WHEN THE TIME LIMIT IS 120S

Instance			The cooperative algorithm			GVNS		
$name$	n	m	Minimal	Average	Maximal	Minimal	Average	Maximal
pr01	48	4	$\geq$	$\geq$	$\geq$	2.9	3.6	4.4
pr02	96	4	$\geq$	$\geq$	$\geq$	15.2	25.6	34.0
pr03	144	4	16.0	50.6	$\geq$	30.9	57.2	75.3
pr04	192	4	$\geq$	$\geq$	$\geq$	61.4	137.2	196.9
pr05	240	4	$\geq$	$\geq$	$\geq$	134.3	208.4	252.0
pr06	288	4	$\geq$	$\geq$	$\geq$	216.0	309.9	363.0
pr07	72	4	77.0	102.4	$\geq$	14.1	17.9	22.3
pr08	144	4	$\geq$	$\geq$	$\geq$	45.3	67.3	93.7
pr09	216	4	26.0	76.2	$\geq$	149.3	171.1	186.3
pr10	288	4	$\geq$	$\geq$	$\geq$	246.6	275.1	310.1

$\geq$ means that the time consumed is not smaller than 120 seconds

Algorithm 4.

IV. EXPERIMENTAL STUDY

The cooperative algorithm was coded in C++ and tested on a PC with Pentium IV 3.0GHz CPU and 2GB RAM. The linear relaxation of the restricted master problem is solved by CPLEX. It was tested on ten instances, called pr01-pr10, which are available at http://www.mech.kuleuven.be/en/cib/op. The parameter $NoImp$ and k_{max} were set to 20 and 9 respectively. Each instance was tested 5 times as done in [12].

We compared the cooperative algorithm with the granular neighborhoods based VNS algorithm (GVNS) [12] since it is effective and the details of experimental results are available. GVNS was tested on a PC Pentium(R)IV with 3.0 GHZ CPU.

In the experiments, our algorithm was stopped when the time limit is reached even when the list of nodes is not empty.

A. The time limit is 120s

Table I reports the results obtained by these algorithms. Column 1-3 present the information of each instance including name, the number of nodes and the number of vehicles. Column 4 presents the best known results obtained so far ($Best\ known$). Column 5-7 report the worst, average, and best values of the cooperative algorithm. Column 8-10 report the worst, average, and best values of the GVNS.

In terms of the best value, our algorithm can find 3 new better results. With respect to the average value, our algorithm is better than GVNS on 6 out of 10 instances.

As seen from Table II, in terms of the average time, our algorithm consumes less time than GVNS on 6 instances and consumes more time on 4 instances.

B. The time limit is 600s

We further study the performance by setting the time limit as 600s. The results obtained by our algorithm are compared with the ones obtained by GVNS in Table III.

In terms of the best value, our algorithm can find 4 new better results, and it finds the same value for 2 instances. With respect to the average value, the cooperative algorithm is better than GVNS on 9 out of 10 instances.

The running times of these algorithms are reported in Table IV. It can be seen that our algorithm consumes more time than GVNS except pr03.

V. CONCLUSION

This paper has introduced a cooperative algorithm which combines metaheuristic and branch-and-price for the team orienteering problem with time windows. In contrast to the exact version of branch-and-price, the proposed algorithm uses a VNS for generating new columns, and uses a heuristic to obtain an integral solution. Based on experimental study, this algorithm can find new better solutions for several instances, which shows that it is potential to design algorithms inspired by the ideas of branch-and-price and metaheuristic. However, we notice that the proposed algorithm may require more running time. In the future, we plan to use some acceleration strategies to alleviate this limitation, such as, using node selection heuristic [19] and adopting other metaheuristics for generating new columns.

ACKNOWLEDGMENT

This work was supported in part by the Open Research Fund of the State Key Laboratory of Astronautic Dynamics under Grant 2014ADL-DW0402, and China Postdoctoral Science Foundation funded project 201003675.

TABLE III. THE RESULTS ON THE TOTAL REWARD OBTAINED FOR THE TEN INSTANCES WHEN THE TIME LIMIT IS 600S

Instance			$Best\ known$	The cooperative algorithm			GVNS		
$name$	n	m		Worst	Average	Best	Worst	Average	Best
pr01	48	4	657	647	653.6	*657*	654	655.8	*657*
pr02	96	4	1079	1067	1068.8	1071	1055	1064.2	1070
pr03	144	4	1232	1234	1240.2	**1246**	1202	1209.6	1215
pr04	192	4	1585	1560	1568.0	1575	1522	1525.8	1531
pr05	240	4	1838	1804	1826.8	**1844**	1785	1811.4	1827
pr06	288	4	1855	1856	1867.2	**1886**	1826	1840.4	1855
pr07	72	4	876	875	875.8	*876*	855	862.6	870
pr08	144	4	1382	1362	1374.0	**1385**	1355	1358.2	1361
pr09	216	4	1619	1601	1611.0	1618	1589	1595.4	1607
pr10	288	4	1943	1907	1911.6	1914	1894	1903.6	1916

Boldface indicates a new best result is obtained
Italic indicates the same best-so-far results is obtained

TABLE IV. THE RUNNING TIME (IN SECONDS) CONSUMED BY THE COMPARED ALGORITHMS FOR THE TEN INSTANCES WHEN THE TIME LIMIT IS 600S

Instance			The cooperative algorithm			GVNS		
$name$	n	m	Minimal	Average	Maximal	Minimal	Average	Maximal
pr01	48	4	180.0	394.6	$\geq$	2.9	3.6	4.4
pr02	96	4	$\geq$	$\geq$	$\geq$	15.2	25.6	34.0
pr03	144	4	9.0	27.4	53.0	30.9	57.2	75.3
pr04	192	4	$\geq$	$\geq$	$\geq$	61.4	137.2	196.9
pr05	240	4	$\geq$	$\geq$	$\geq$	134.3	208.4	252.0
pr06	288	4	$\geq$	$\geq$	$\geq$	216.0	309.9	363.0
pr07	72	4	66.0	93.4	125.0	14.1	17.9	22.3
pr08	144	4	$\geq$	$\geq$	$\geq$	45.3	67.3	93.7
pr09	216	4	56.0	388.8	$\geq$	149.3	171.1	186.3
pr10	288	4	$\geq$	$\geq$	$\geq$	246.6	275.1	310.1

$\geq$ means that the time consumed is not smaller than 600 seconds

REFERENCES

[1] R. Montemanni and L. Gambardella, "An ant colony system for team orienteering problems with time windows," *Foundations of Computing and Decision Sciences*, vol. 34, pp. 287–306, 2009.

[2] I. Chao, B. Golden, and E. Wasil, "The team orienteering problem," *European Journal of Operational Research*, vol. 88, pp. 464–474, 1996.

[3] S. Boussier, D. Feillet, and M. Gendreau, "An exact algorithm for team orienteering problems," *4OR: A Quarterly Journal of Operations Research*, vol. 5, no. 3, pp. 211–230, 2007.

[4] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[5] R. Montemanni, D. Weyland, and L. Gambardella, "An enhanced ant colony system for the team orienteering problem with time windows," in *the 2011 International Symposium on Computer Science and Society (ISCCS,2011)*, Kota Kinabalu,Malaysia, July 2011, pp. 381–384.

[6] L. M. Gambardella, R. Montemanni, and D. Weyland, "Coupling ant colony systems with strong local searches," *European Journal of Operational Research*, vol. 220, no. 3, pp. 831–843, 2012.

[7] F. Tricoire, M. Romauch, K. F. Doerner, and R. F. Hartl, "Heuristics for the multi-period orienteering problem with multiple time windows," *Computers & Operations Research*, vol. 37, no. 2, pp. 351–367, 2010.

[8] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden, "Iterated local search for the team orienteering problem with time windows," *Computers & Operations Research*, vol. 36, no. 12, pp. 3281–3290, 2009.

[9] W. Souffriau, P. Vansteenwegen, G. V. Berghe, and D. Van Oudheusden, "The multiconstraint team orienteering problem with multiple time windows," *Transportation Science*, vol. 47, no. 1, pp. 53–63, 2013.

[10] N. Labadi, J. Melechovsky, and R. Calvo, "An effective hybrid evolutionary local search for orienteering and team orienteering problems with time windows," in *Parallel Problem Solving from Nature, PPSN XI*, ser. Lecture Notes in Computer Science, R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, Eds. Springer, 2010, vol. 6239, pp. 219–228.

[11] S. W. Lin and V. F. Yu, "A simulated annealing heuristic for the team orienteering problem with time windows," *European Journal of Operational Research*, vol. 217, no. 1, pp. 94–107, 2012.

[12] N. Labadie, R. Mansini, J. Melechovskỳ, and R. Wolfler Calvo, "The team orienteering problem with time windows: An LP-based granular variable neighborhood search," *European journal of operational research*, vol. 220, no. 1, pp. 15–27, 2012.

[13] Q. Hu and A. Lim, "An iterative three-component heuristic for the team orienteering problem with time windows," *European Journal of Operational Research*, 2013.

[14] K. Karabulut and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the team orienteering problem with time windows," in *the 2013 IEEE Workshop on Computational Intelligence In Production And Logistics Systems (CIPLS,2013)*, Singapore,Singapore, April 2013, pp. 99–106.

[15] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and Y. Tasoulas, "Cluster-based heuristics for the team orienteering problem with time windows," in *Experimental Algorithms*, ser. Lecture Notes in Computer Science, V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, Eds. Springer, 2013, vol. 7933, pp. 390–401.

[16] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[17] E. G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *4OR: A Quarterly Journal of Operations Research*, vol. 11, pp. 101–150, 2013.

[18] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European journal of operational research*, vol. 130, no. 3, pp. 449–467, 2001.

[19] M. Fischetti and A. Lodi, "Local branching," *Mathematical Programming*, vol. 98, no. 1-3, pp. 23–47, 2003.