symmetry

MDPI

Brief Report

On Solving the Set Orienteering Problem

Roberto Montemanni [1,*] and Derek H. Smith [2]

[1] Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia, RE, Italy
[2] Computing and Mathematics, University of South Wales, Pontypridd CF37 1DL, Wales, UK; derek.smith@southwales.ac.uk
* Correspondence: roberto.montemanni@unimore.it; Tel.: +39-0522-522-126

Abstract: In the Set Orienteering Problem, a single vehicle, leaving from and returning to a depot, has to serve some customers, each one associated with a given spacial location. Customers are grouped in clusters and a given prize is collected once a customer in a cluster is visited. The prize associated with a cluster can be collected at most once. Travel times among locations are provided, together with a maximum available mission time, which normally makes it impossible to visit all the clusters. The target is to design a route for the vehicle that maximizes the total prize collected within the given time limit. In this study, building on the recent literature, we present new preprocessing rules and a new constraint programming model for the problem. Thanks to the symmetry exploitation carried out by the constraint programming solver, new state-of-the-art results are established.

Keywords: set orienteering problem; constraint programming; preprocessing rules; exact methods

1. Introduction

The Orienteering Problem (OP) was introduced in [1]. In the problem, a single vehicle, leaving from and returning to a depot, serves a set of customers, each one associated with a spacial location and a prize, which is collected upon visit. Travel times among locations are provided. Not all the customers can typically be serviced, since the vehicle mission cannot be longer than a given maximum time. The aim is to maximize the total profit collected by the vehicle in the given available time. The problem has attracted a lot of attention due to its practical implications, and many variations of the original problem have been introduced over the years. We refer the interested reader to [2] for an exhaustive review of the literature on these problems.

The problem addressed in the present study is the Set Orienteering Problem (SOP), which was introduced in [3], where customers are grouped in (non-overlapping) clusters and a profit is associated with each cluster. Such a profit is collected if at least one customer in the cluster is visited. The problem is not to be confused with the Sequential Ordering Problem [4], which has been abbreviated with the same acronym for much longer.

The SOP has several real applications. It can be used to model situations where a carrier delivers goods to a company with multiple warehouses, and the delivery can be carried out to one of them. Another important application that emerged recently is in last-mile delivery: when the delivery to a customer can be made in different locations (for example, home, work, pickup station, or delivery locker), the carrier can choose the most convenient one. There is a flourishing body of literature for these applications, where the most disparate realistic constraints are added to the basic problem (see, for example, [5,6]).

Heuristic methods for the SOP, targeting instances of any practical size, were introduced in [7,8]. The former method is based on variable neighbour search, while the latter implements a biased random-key genetic algorithm. Exact methods, targeting small-/medium-size instances only, but with the advantage of providing upper bounds in addition to feasible solutions, were proposed in [3,7]. Very recently, a more elaborate Branch-and-Cut method, representing the current state of the art for exact algorithms, was discussed in [9].

Citation: Montemanni, R.; Smith, D.H. On Solving the Set Orienteering Problem. *Symmetry* **2024**, *16*, 340. https://doi.org/10.3390/sym16030340

Academic Editors: Jianchao Bai, Qiyu Jin and Yuchao Tang

Received: 9 February 2024
Revised: 8 March 2024
Accepted: 11 March 2024
Published: 12 March 2024

The present work provides three main contributions. First, a new effective preprocessing rule is introduced, able to substantially reduce the size of the instances by identifying and removing vertices that cannot be part of any feasible solution. Second, a new constraint programming (CP) model, following the same approach recently proposed for other problems [10], is introduced. Third, by combining the previous two contributions, new state-of-the-art results are obtained. A main factor that led to such an achievement is the heavy symmetry exploitation carried out by the CP solver adopted (see [11–14]).

2. Problem Description

Let $G = (V, A)$ be a complete digraph, where $V = \{0\} \cup C$. The depot (starting and ending point of the route) is vertex 0, while C is the set of customers. Customers in C are partitioned into clusters $C_0, C_1, \ldots, C_m$. A profit p_g is associated to each cluster C_g and such a profit is collected if at least one customer $i \in C_g$ is visited. Cluster C_0 contains only the depot 0 and has a null profit. A travel time c_{ij} is associated with each arc $(i, j) \in A$, and a maximum time T_{max} is given. The Set Orienteering Problem (SOP) consists of finding a route no longer than T_{max} that maximizes the profit collected. A simplified example of an SOP instance and the relative solution is provided in Figure 1.

In the remainder of the paper, we assume—consistently with the previous literature—that the travel times c satisfy the triangle inequality. This implies that an optimal solution containing at most one vertex for each cluster exists. As a consequence, arcs between vertices of a single cluster can be removed from the graph.

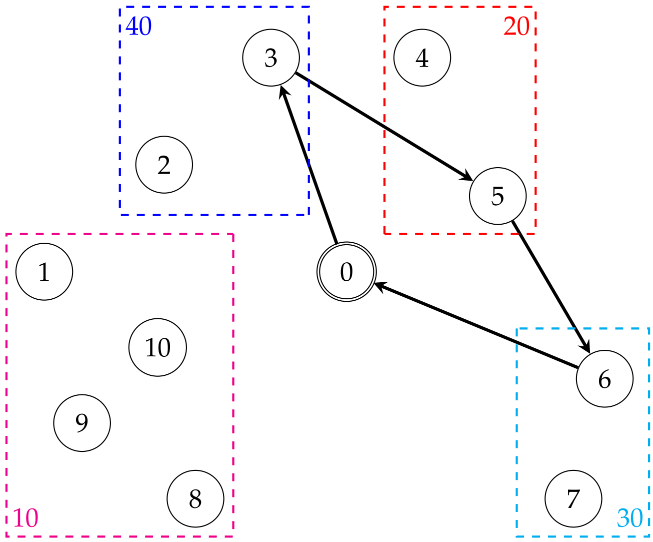

Figure 1. Example of an SOP instance. Node 0 is the depot, while the other nodes are customers. Clusters are represented as coloured rectangles, with the associated prize depicted in a corner. Travel times are omitted for the sake of simplicity, together with the threshold T_{max}. A tour with a total prize of 90 is drawn in black.

3. Preprocessing Rules

Some preprocessing techniques, with the function of reducing the number of variables and edges, are introduced in this section. We refer the interested reader to [8] for a more detailed explanation of Theorems 1 and 2 and for their proofs.

Theorem 1 (Carrabs [8])**.** *Given a cluster C_g, let S be the set of the shortest paths from every $u \in C_h$ to every $v \in C_k$ passing through C_g, with $h \neq k \neq g$. Moreover, let A_S be the set of arcs incident to the vertices in C_g that do not belong to any shortest path of S. An optimal solution not containing arcs in A_S always exists. The arcs in A_S can be removed from the graph.*

Theorem 2 (Carrabs [8])**.** *Given a cluster C_g, let S be the set of the shortest paths from every $u \in C_h$ to every $v \in C_k$ passing through C_g, with $h \neq k \neq g$. Moreover, let V_S be the set of vertices in C_g that do not belong to any shortest path of S. An optimal solution of the SOP not containing vertices in V_S always exists. The vertices in V_S can be removed from the graph.*

Theorem 3. *Let $SP(i,j)$ be the cost of the shortest path from vertex $i \in G$ to vertex $j \in G$. Given $k \in V$, if $SP(0,k) + SP(k,0) > T_{max}$, then the vertex k cannot be part of any feasible solution and can be removed from the graph.*

Proof. If vertex k is only part of vehicle routes longer than T_{max}, then no feasible solution with k exists and it can be eliminated from the graph. □

Remark 1. *In case the arc (i,j) exists in the graph, $SP(i,j) = d_{ij}$ due to triangle inequalities. Otherwise, an alternative path might exist, and it needs to be calculated explicitly.*

In our implementation, the three theorems are applied sequentially within a loop, which is executed until no further reduction is possible.

4. A Constraint Programming Model

The SOP can be described through the following constraint programming model, designed according to the syntax of the Google OR-Tools CP-SAT solver [14]. Given a vertex $i \in V$, we will indicate with $cl(i)$ the unique cluster containing i. A binary variable x_{ij}, with $i,j \in V$, takes value 1 if vertex i is visited right before vertex j in the solution tour, and value 0 otherwise. In case a customer $i \in C$ is not visited, then x_{ii} is set to 1, and 0 otherwise.

$$\max \sum_{i \in V} p_{cl(i)} \sum_{j \in V, j \neq i} x_{ij} \tag{1}$$

$$s.t. \quad AddCircuit(x_{ij}; i,j \in V; i \neq 0 \vee j \neq 0) \tag{2}$$

$$\sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij} x_{ij} \leq T_{max} \tag{3}$$

$$x_{ij} \in \{0;1\} \qquad i,j \in V \tag{4}$$

The objective function (1) maximizes the profit collected in the tour. Constraint (2) imposes that the tour associated with the active x variables forms a feasible circuit. This is imposed by the CP-SAT statement *AddCircuit* that also ensures that $x_{ii} = 1$ for each variable $i \in C$ not touched by the circuit itself. The constraints will ensure that only solutions in the shape of a tour will be considered, and, combined with the objective function (1) and the following constraints (3), will guarantee that only feasible solutions are generated. Constraint (3) is a budget constraint requiring that the length of the tour described by the active x variables has a length of at most T_{max}. Notice that the critical values of T_{max} that will make the optimization harder are those in the medium range: small values would lead to an easy problem because just a few vertices could be selected and, conversely, large values would take the problem closer to a traditional Traveling Salesman Problem, with the selection of just a few vertices to be left out. Constraints (4) finally define the domain of the variables.

The following constraints are added to tighten the model, although they would not be required:

$$AddAtMostOne(\neg x_{ii}; i \in C_g) \qquad g \in \{1,2,\ldots,l\} \tag{5}$$

Constraints (5) impose that for each cluster g at most one customer is selected, since every optimal solution will respect this property due to the distances fulfilling triangle inequalities. The constraints are based on the use of the *AddAtMostOne* of CP-SAT and the negation operator "¬" (*Not* in CP-SAT). These constraints are included for all the experiments reported in Section 5, since they contribute to speeding up the solving process.

5. Computational Experiments

The computational tests were carried out on the instances previously adopted in the literature on exact algorithms. Two sets of instances were introduced in [3], for a total of 228 instances. *Set1* is composed of instances with a number of vertices between 52 and 198. The parameter ω, taking values 0.4, 0.6, and 0.8, regulates the value of T_{max}. Two different rules, $g1$ and $g2$, are finally used to assign the profit to the clusters. The instances of *Set2* contain the same vertices and the same number of clusters as those in *Set1*, but the vertices are assigned in a different way to the clusters. We refer the interested reader to [3] for the full details of these instances.

In Table 1, we report statistics about the preprocessing procedures used in [8]—employing Theorems 1 and 2 only—and the full methodology we propose, which also uses Theorem 3. All the procedures were implemented from scratch in Python and the results reported were obtained on a computer equipped with an Intel Core i7 12700F processor and 32 GB of RAM. For each procedure, we considered the percentage of dominated nodes, the percentage of dominated arcs, and the computation time required. For each of these indicators, we report the minimum, maximum, and average values over the 228 instances considered.

Table 1. Preprocessing performance. Statistics over the 228 instances considered.

		Theorems 1 and 2 (Carrabs [8])	**Theorems 1–3**
Dominated nodes (%)	Min	0.00	0.00
	Max	13.00	65.66
	Avg	2.46	20.18
Dominated arcs (%)	Min	19.02	22.98
	Max	68.93	93.71
	Avg	40.76	57.88
Computation time (s)	Min	0.26	0.05
	Max	153.15	23.04
	Avg	11.26	3.61

When Theorem 3 is considered, the percentage of dominated nodes increases substantially, together with the percentage of dominated arcs (although in a weaker form). In particular, looking at the *Min* and *Max* values, it appears that some instances benefit substantially from the new reduction. Looking at the computation times required by preprocessing, a remarkable reduction is associated with the use of Theorem 3, which eventually leads to an early identification of dominated elements. Also in this case, the impressive gain in the *Max* row suggests that there are instances very sensitive to Theorem 3. The success of Theorem 3 as a preprocessing method can be explained by observing that it is the first method to take into account T_{max}, the maximum travel time allowed for the tour of the truck, and travel times. In the economy of the problem, this is an important factor, since the results often show the existence of several vertices that cannot simply be visited in the given time. Moreover, it must be observed how Theorem 3 builds on the results on the other theorems, and in turns boost them back. However, the results remain dependent on the characteristics of each instances, and this explains the fluctuations in the results achieved.

In Tables 2–5, we compare the method proposed in this paper with the existing approaches from the literature. We consider the Mixed Integer Program (MIP) from [3] (*clucut*), solved as described in [9], and the Branch-and-Cut method introduced in [9] (*BC*). For these methods, we report the results published in the literature, with Theorem 1 and 2 used for preprocessing. Conversely, the constraint programming model discussed in Section 4 (*CP*) uses all the results discussed in Section 3 for preprocessing. For each of the three methods, the cost of the best solution found in the time allowed, the time required to eventually prove optimality and the eventual optimality gap (calculated as $(UB - LB)/LB$, where UB and LB are the best upper and lower bounds returned by the solver, respectively), are reported for each instance. The maximum time allowed (also considering preprocessing) is 3600 s on an Intel Core i9-10910 3.6 GHz processor with 64 GB of RAM for *clucut* and *BC*, and 36,000 s on a Intel Core i7 2.1 GHz processor with 32 GB of RAM for *CP*. We decided to extend the time allowed to *CP* in the hope of closing more instances. CP-SAT 9.8 [14] with standard settings has previously been adopted as a solver for constraint programming models.

The results are clearly in favour of the *CP* method (combined with the use of Theorem 3) for all the instances considered, both in terms of average computation times and solution quality. Only two instances remain open, namely, *22pr107* with $\omega = 0.8$ and *Set1* for both profit rules g_1 and g_2. The dominating results depend both on the new preprocessing rule described in Theorem 3 and on the effectiveness of the solver for the constraint programming model discussed in Section 4, which—as observed in Section 1—depends strongly on symmetry exploitation carried out by the solver itself, as documented in [14]. Some tests not reported—as the aim of this report is mainly to present the new state-of-the-art results—indicated that the new preprocessing rule and the efficiency of the constraint programming model on the new model both contributed to the results obtained. Notice in particular that CP-SAT models being faster to solve than traditional MIPs is consistent with results recently presented for other similar vehicle routing-like problems [10].

Table 2. Experimental results on the instances from *Set1* with $\omega = 0.4$.

	Instance	g_1									g_2								
		***clucut* (Archetti et al. [9])**			***BC* (Archetti et al. [9])**			***CP***			***clucut* (Archetti et al. [9])**			***BC* (Archetti et al. [9])**			***CP***		
		Val	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**
Set1	11berlin52	37	0.6	0.0	37	0.5	0.0	37	0.7	0.0	1829	0.6	0.0	1829	0.5	0.0	1829	0.6	0.0
	11eil51	24	0.1	0.0	24	0.2	0.0	24	1.1	0.0	1279	0.3	0.0	1279	1.2	0.0	1279	1.0	0.0
	14st70	33	0.5	0.0	33	0.5	0.0	33	1.4	0.0	1672	0.6	0.0	1672	0.5	0.0	1672	1.7	0.0
	16eil76	40	3.9	0.0	40	2.6	0.0	40	6.2	0.0	2223	3.8	0.0	2223	1.8	0.0	2223	9.0	0.0
	16pr76	47	5.0	0.0	47	5.2	0.0	47	11.7	0.0	2449	15.3	0.0	2449	4.0	0.0	2449	13.3	0.0
	20kroA100	42	49.1	0.0	42	29.0	0.0	42	75.3	0.0	2151	84.6	0.0	2151	32.6	0.0	2151	160.6	0.0
	20kroB100	49	15.8	0.0	49	8.9	0.0	49	26.1	0.0	2431	28.0	0.0	2431	16.1	0.0	2431	30.0	0.0
	20kroC100	42	3.1	0.0	42	2.1	0.0	42	6.9	0.0	2174	3.1	0.0	2174	4.6	0.0	2174	11.8	0.0
	20kroD100	39	3.4	0.0	39	3.1	0.0	39	7.1	0.0	1740	9.6	0.0	1740	8.5	0.0	1740	40.0	0.0
	20kroE100	52	3.7	0.0	52	3.9	0.0	52	8.0	0.0	2415	2.6	0.0	2415	5.3	0.0	2415	17.4	0.0
	20rat99	37	0.9	0.0	37	1.7	0.0	37	2.0	0.0	1905	0.8	0.0	1905	0.6	0.0	1905	2.0	0.0
	20rd100	45	6.6	0.0	45	7.9	0.0	45	19.3	0.0	2228	13.6	0.0	2228	7.4	0.0	2228	49.1	0.0
	21eil101	67	48.9	0.0	67	12.6	0.0	67	62.2	0.0	3365	61.4	0.0	3365	15.9	0.0	3365	152.9	0.0
	21lin105	50	16.9	0.0	50	32.5	0.0	50	6.8	0.0	2489	13.3	0.0	2489	13.5	0.0	2489	11.5	0.0
	22pr107	41	0.0	0.0	41	0.0	0.0	41	0.2	0.0	2123	0.1	0.0	2123	0.1	0.0	2123	0.2	0.0
	25pr124	46	2375.0	0.0	46	114.7	0.0	46	494.7	0.0	2302	3635.9	32.8	2302	182.2	0.0	2302	1328.2	0.0
	26bier127	109	3761.2	8.6	110	1002.4	0.0	110	257.6	0.0	5069	3686.8	15.5	5420	2991.9	0.0	5420	860.5	0.0
	26ch130	67	3752.9	28.5	70	371.6	0.0	70	2638.8	0.0	3320	3747.8	26.5	3423	820.3	0.0	3423	6863.9	0.0
	28pr136	53	286.1	0.0	53	33.2	0.0	53	5938.8	0.0	2699	449.7	0.0	2699	327.5	0.0	2699	4506.2	0.0
	29pr144	6	3663.4	94.1	60	1739.5	0.0	60	2690.1	0.0	3055	3774.9	39.2	3055	1707.9	0.0	3055	2231.4	0.0
	30ch150	61	3741.8	21.0	61	536.1	0.0	61	5113.6	0.0	3078 *	3527.5	0.0	3078 *	749.8	0.0	3131	7300.5	0.0
	30kroA150	58	3748.0	30.9	58	654.2	0.0	58	2919.2	0.0	3026	3739.4	18.2	3039	779.7	0.0	3039	2316.5	0.0
	30kroB150	66	3722.5	16.8	66	354.7	0.0	66	8119.6	0.0	3172	3731.6	24.7	3172	2081.3	0.0	3172	10,963.6	0.0
	31pr152	9	3653.2	91.4	57	949.2	0.0	57	2841.2	0.0	2440	3651.9	54.7	2915	1574.6	0.0	2915	3000.1	0.0
	32u159	76	1791.0	0.0	76	1429.4	0.0	76	2336.9	0.0	4002	2568.6	0.0	4002	584.4	0.0	4002	2838.6	0.0
	39rat195	71	1354.3	0.0	71	311.4	0.0	71	2850.2	0.0	3656	1034.4	0.0	3656	287.2	0.0	3656	3416.1	0.0
	40d198	67 *	181.3	0.0	67 *	85.9	0.0	70	502.2	0.0	3400 *	229.7	0.0	3400 *	49.0	0.0	3595	929.5	0.0
	Average	**49.4**	**1192.2**	**10.8**	**53.3**	**284.9**	**0.0**	**53.4**	**1368.1**	**0.0**	**2655.3**	**1259.8**	**7.8**	**2690.1**	**453.6**	**0.0**	**2699.3**	**1742.8**	**0.0**

[*] This cost is not consistent with the results of *CP* or with the results reported in [7,8] for some heuristic approaches. As a consequence, the entry in the table of [9] requires recalculation and update [15].

Table 3. Experimental results on the instances from *Set2* with $\omega = 0.4$.

	Instance	g_1									g_2								
		clucut (Archetti et al. [9])			*BC* (Archetti et al. [9])			*CP*			*clucut* (Archetti et al. [9])			*BC* (Archetti et al. [9])			*CP*		
		Val	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**
Set2	11berlin52	50	1.0	0.0	50	1.0	0.0	50	0.5	0.0	2584	0.8	0.0	2584	0.9	0.0	2584	0.6	0.0
	11eil51	37	0.3	0.0	37	2.5	0.0	37	1.3	0.0	1929	0.2	0.0	1929	0.6	0.0	1929	3.7	0.0
	14st70	56	2.0	0.0	56	0.8	0.0	56	5.3	0.0	2736	1.8	0.0	2736	0.9	0.0	2736	7.5	0.0
	16eil76	51	4.1	0.0	51	2.7	0.0	51	19.2	0.0	2518	6.8	0.0	2518	11.8	0.0	2518	44.6	0.0
	16pr76	70	161.9	0.0	70	156.1	0.0	70	29.4	0.0	3550	146.1	0.0	3550	33.1	0.0	3550	19.3	0.0
	20kroA100	80	1478.1	0.0	80	42.4	0.0	80	139.3	0.0	3894	848.4	0.0	3894	56.7	0.0	3894	205.5	0.0
	20kroB100	86	664.7	0.0	86	52.4	0.0	86	46.3	0.0	4357	678.8	0.0	4357	433.0	0.0	4357	56.0	0.0
	20kroC100	72	132.1	0.0	72	28.2	0.0	72	169.8	0.0	3586	206.2	0.0	3586	99.6	0.0	3586	398.8	0.0
	20kroD100	78	28.3	0.0	78	11.0	0.0	78	51.2	0.0	3799	112.8	0.0	3799	33.4	0.0	3799	51.7	0.0
	20kroE100	90	191.2	0.0	90	8.0	0.0	90	19.7	0.0	4614	25.4	0.0	4614	28.7	0.0	4614	19.3	0.0
	20rat99	73	0.3	0.0	73	1.7	0.0	73	2.9	0.0	3624	1.1	0.0	3624	43.5	0.0	3624	8.2	0.0
	20rd100	80 *	44.4	0.0	80 *	26.6	0.0	82	89.4	0.0	4038 *	34.1	0.0	4038 *	47.1	0.0	4181	163.3	0.0
	21eil101	83	47.7	0.0	83	31.1	0.0	83	245.1	0.0	4264	72.8	0.0	4264	48.0	0.0	4264	451.2	0.0
	21lin105	95	753.1	0.0	95	378.5	0.0	95	117.8	0.0	4814	879.2	0.0	4814	403.0	0.0	4814	156.5	0.0
	22pr107	94	10.6	0.0	94	14.3	0.0	94	7.6	0.0	4740	76.3	0.0	4740	20.2	0.0	4740	4.4	0.0
	25pr124	90	3625.7	25.6	101	832.8	0.0	101	1831.8	0.0	4334	3622.9	28.4	3859	3625.2	36.3	5035	3501.1	0.0
	26bier127	11	3656.2	91.3	124	3656.5	1.6	125	78.5	0.0	6236	3673.1	1.5	6004	3637.1	5.2	6329	176.0	0.0
	26ch130	9	3622.2	93.0	9	3632.6	92.9	111	3193.2	0.0	153	3625.7	97.6	4833	3633.2	24.4	5630	20,566.6	0.0
	28pr136	120	2524.2	0.0	120	37.8	0.0	120	134.7	0.0	6106	1789.0	0.0	6106	157.3	0.0	6106	367.4	0.0
	29pr144	4	3637.3	97.2	4	3630.0	97.2	137	754.9	0.0	166	3628.3	97.7	166	3626.4	97.7	6848	1591.4	0.0
	30ch150	90	3627.8	39.6	111 *	1524.7	0.0	114	2501.2	0.0	4361	3633.8	42.1	5896 *	2552.9	0.0	6025	1155.1	0.0
	30kroA150	11	3626.8	92.6	99	3634.2	33.6	110	10,533.2	0.0	141	3626.6	98.1	4478	3636.8	39.9	5450	12,838.2	0.0
	30kroB150	9	3631.1	93.9	115	3630.1	22.0	120	13,969.2	0.0	171	3627.9	97.7	6190	3624.8	17.7	6255	15,700.5	0.0
	31pr152	89	3632.4	40.7	9	3629.2	94.0	136	30,240.8	0.0	431	3636.3	94.3	431	3630.9	94.3	6928	8101.4	0.0
	32u159	143	3627.6	7.1	143	428.1	0.0	143	565.1	0.0	7507	3620.3	4.4	7507	913.5	0.0	7507	464.2	0.0
	39rat195	135	740.9	0.0	135	467.6	0.0	135	244.0	0.0	6813	1190.8	0.0	6813	288.8	0.0	6813	485.7	0.0
	40d198	148 *	844.7	0.0	148 *	178.1	0.0	149	1192.9	0.0	7412 *	1082.8	0.0	7412 *	393.3	0.0	7480	2082.2	0.0
	Average	**72.4**	**1493.2**	**21.5**	**82.0**	**964.4**	**12.6**	**96.2**	**2451.3**	**0.0**	**3662.1**	**1475.9**	**20.8**	**4249.7**	**1147.4**	**11.7**	**4873.9**	**2541.5**	**0.0**

[*] This cost is not consistent with the results of *CP* or with the results reported in [7,8] for some heuristic approaches. As a consequence, the entry in the table of [9] requires recalculation and update [15].

Table 4. Experimental results on the instances with $\omega = 0.6$.

	Instance	g_1									g_2								
		***clucut* (Archetti et al. [9])**			***BC* (Archetti et al. [9])**			***CP***			***clucut* (Archetti et al. [9])**			***BC* (Archetti et al. [9])**			***CP***		
		Val	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**
	11eil51	39	0.6	0.0	39	1.9	0.0	39	1.3	0.0	1911	0.6	0.0	1911	1.2	0.0	1911	4.1	0.0
	14st70	50	101.2	0.0	50	21.7	0.0	50	63.8	0.0	2589	39.2	0.0	2589	20.7	0.0	2589	112.1	0.0
	16eil76	59	76.9	0.0	59	8.9	0.0	59	10.1	0.0	3119	82.0	0.0	3119	21.5	0.0	3119	24.0	0.0
	16pr76	65	69.8	0.0	65	133.8	0.0	65	207.4	0.0	3275	1496.7	0.0	3275	190.7	0.0	3275	2286.2	0.0
	20kroA100	65	1979.6	0.0	65	110.0	0.0	65	177.0	0.0	3192	1740.7	0.0	3192	140.0	0.0	3192	1088.5	0.0
	20kroB100	59	3628.8	39.8	66	100.8	0.0	66	1161.2	0.0	3203	1966.9	0.0	3203	167.7	0.0	3203	1713.8	0.0
	20kroC100	62	521.3	0.0	62	74.9	0.0	62	575.0	0.0	3110	1700.9	0.0	3110	255.1	0.0	3110	876.1	0.0
	20kroD100	64	2517.9	0.0	64	78.3	0.0	64	438.6	0.0	3133	2324.2	0.0	3133	84.4	0.0	3133	473.8	0.0
	20kroE100	63	107.7	0.0	63	190.1	0.0	63	146.9	0.0	2950	318.5	0.0	2950	89.8	0.0	2950	324.5	0.0
	20rat99	52	130.3	0.0	52	50.5	0.0	52	185.0	0.0	2643	80.6	0.0	2643	44.1	0.0	2643	383.7	0.0
	20rd100	72	450.0	0.0	72	67.1	0.0	72	186.9	0.0	3585 *	413.5	0.0	3585 *	278.7	0.0	3591	901.9	0.0
	21eil101	82	913.4	0.0	82	85.7	0.0	82	261.5	0.0	4187	720.2	0.0	4187	447.3	0.0	4187	1657.9	0.0
	21lin105	78	504.6	0.0	78	137.8	0.0	78	82.9	0.0	3955	1178.4	0.0	3955	171.1	0.0	3955	197.2	0.0
	22pr107	53	3623.0	36.1	53	3624.2	31.2	53	30.6	0.0	2697	3626.8	34.7	2697	3627.4	30.4	2697	127.9	0.0
	Average	**60.4**	**975.2**	**5.1**	**60.9**	**312.7**	**2.1**	**60.9**	**235.4**	**0.0**	**3049.3**	**1046.1**	**2.3**	**3049.3**	**369.6**	**2.0**	**3049.7**	**678.4**	**0.0**
Set2	11berlin52	51	0.1	0.0	51	0.1	0.0	51	0.5	0.0	2608	0.1	0.0	2608	0.2	0.0	2608	0.5	0.0
	11eil51	50	0.6	0.0	50	3.8	0.0	50	0.9	0.0	2575	0.6	0.0	2575	0.6	0.0	2575	0.9	0.0
	14st70	64	2152.4	0.0	64	341.9	0.0	64	979.2	0.0	3218	3619.3	8.4	3218	569.4	0.0	3218	815.3	0.0
	16eil76	74	526.1	0.0	74	193.9	0.0	74	8.2	0.0	3728	117.1	0.0	3728	108.6	0.0	3728	26.7	0.0
	16pr76	74	3619.8	1.3	74	2088.3	0.0	74	12.5	0.0	3729	3621.3	1.9	3729	532.5	0.0	3729	36.5	0.0
	20kroA100	91	3624.9	8.1	95	3624.0	4.0	98	533.0	0.0	3763	3630.4	24.9	4554	3621.7	9.1	4920	912.6	0.0
	20kroB100	93	3628.7	6.1	2	3621.6	98.0	98	2087.6	0.0	3578	3630.6	28.6	4668	3624.1	6.8	4925	390.7	0.0
	20kroC100	5	3625.9	94.9	90	3620.4	9.1	93	11,210.5	0.0	3915	3622.6	21.8	4534	3618.5	9.5	4717	2482.3	0.0
	20kroD100	4	3623.2	96.0	93	3618.9	6.1	93	2211.4	0.0	4394	3628.4	12.3	4570	3619.6	8.7	4695	2160.7	0.0
	20kroE100	97	3621.6	2.0	97	2619.0	0.0	97	66.0	0.0	4910	3622.6	2.0	4910	3617.8	2.0	4910	93.2	0.0
	20rat99	87	162.3	0.0	87	216.2	0.0	87	118.8	0.0	4516	76.8	0.0	4516	165.9	0.0	4516	76.6	0.0
	20rd100	97	3628.6	2.0	99	3459.8	0.0	99	86.5	0.0	5008	1113.8	0.0	5008	572.6	0.0	5008	12.0	0.0
	21eil101	95	3623.4	5.0	97	1111.3	0.0	97	221.8	0.0	4925	3622.6	2.5	4925	3623.8	2.5	4933	1988.6	0.0
	21lin105	102	3642.8	1.9	104	888.4	0.0	104	32.1	0.0	4495	3631.2	14.0	5103	3627.8	2.4	5228	21.7	0.0
	22pr107	106	243.7	0.0	106	11.2	0.0	106	4.8	0.0	5363	29.3	0.0	5363	139.7	0.0	5363	5.3	0.0
	Average	**72.7**	**2381.6**	**14.5**	**78.9**	**1694.6**	**7.8**	**85.7**	**1171.6**	**0.0**	**4048.3**	**2264.5**	**7.7**	**4267.3**	**1829.5**	**2.7**	**4338.2**	**601.6**	**0.0**

[*] This cost is not consistent with the results of *CP* or with the results reported in [7,8] for some heuristic approaches. As a consequence, the entry in the table of [9] requires recalculation and update [15].

Table 5. Experimental results on the instances with $\omega = 0.8$.

	Instance	g_1									g_2								
		clucut (Archetti et al. [9])			*BC* (Archetti et al. [9])			*CP*			*clucut* (Archetti et al. [9])			*BC* (Archetti et al. [9])			*CP*		
		Val	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**	**Val**	**Sec**	**Gap**
	11eil51	43	4.3	0.0	43	2.2	0.0	43	16.6	0.0	2114	7.4	0.0	2114	7.4	0.0	2114	42.9	0.0
	14st70	65	1110.4	0.0	65	445.8	0.0	65	26.8	0.0	3355	692.9	0.0	3355	661.5	0.0	3355	29.6	0.0
	16eil76	69	695.4	0.0	69	178.5	0.0	69	38.9	0.0	3573	1852.7	0.0	3573	97.0	0.0	3573	65.5	0.0
	16pr76	72	3619.4	2.7	72	1952.1	0.0	72	30.1	0.0	3611	3625.6	3.2	3611	3620.8	2.2	3611	632.0	0.0
	20kroA100	68	3629.7	31.3	79	240.7	0.0	79	1035.2	0.0	2713	3632.4	45.8	4115	3466.8	0.0	4115	2456.3	0.0
	20kroB100	77	3636.5	22.2	86	3125.6	0.0	86	2007.6	0.0	4188	3628.9	16.4	4117	3640.6	16.3	4188	3894.9	0.0
	20kroC100	76	3631.9	23.2	83	466.7	0.0	83	228.6	0.0	3999	3625.9	20.1	3999	300.1	0.0	3999	1423.3	0.0
	20kroD100	68	3635.7	31.3	85	480.4	0.0	85	219.5	0.0	3854	3630.9	23.0	4026	3626.0	19.6	4267	380.6	0.0
	20kroE100	77	3627.7	22.2	80	372.3	0.0	80	1500.6	0.0	3887	3628.6	14.0	4002	414.4	0.0	4002	1281.3	0.0
	20rat99	69	3634.3	21.6	79	2046.6	0.0	79	512.7	0.0	3855	3623.5	13.1	3992	3113.5	0.0	3992	1074.3	0.0
	20rd100	83	3636.9	16.2	90	3629.8	6.3	91	96.7	0.0	4155	3632.6	17.0	4640	1982.4	0.0	4640	102.1	0.0
	21eil101	89	3631.5	11.0	91	347.5	0.0	91	325.0	0.0	4538	3633.8	10.1	4717	1969.2	0.0	4717	615.4	0.0
	21lin105	87	3642.1	16.3	90	302.2	0.0	90	6099.5	0.0	4245	3649.0	18.8	4561	3641.8	10.7	4561	1535.9	0.0
	22pr107	6	3635.6	94.3	53	3650.2	50.0	65	36,000.0	26.2	2156	3638.3	59.8	2697	3636.8	49.7	3275	36,000.0	28.9
	Average	**66.4**	**2788.7**	**19.5**	**74.1**	**1149.8**	**3.8**	**75.0**	**3209.3**	**1.7**	**3508.5**	**2834.3**	**16.1**	**3726.9**	**2012.8**	**6.6**	**3786.2**	**3302.5**	**1.9**
Set2	11berlin52	51	0.0	0.0	51	0.0	0.0	51	0.4	0.0	2608	0.1	0.0	2608	0.1	0.0	2608	0.4	0.0
	11eil51	50	1.4	0.0	50	0.8	0.0	50	0.7	0.0	2575	0.6	0.0	2575	0.5	0.0	2575	0.7	0.0
	14st70	69	8.9	0.0	69	3.7	0.0	69	3.4	0.0	3513	28.8	0.0	3513	14.1	0.0	3513	3.2	0.0
	16eil76	75	14.1	0.0	75	4.1	0.0	75	3.8	0.0	3800	3.7	0.0	3800	177.3	0.0	3800	4.3	0.0
	16pr76	75	2600.9	0.0	75	8.6	0.0	75	5.8	0.0	3800	2501.0	0.0	3800	753.5	0.0	3800	7.5	0.0
	20kroA100	99	395.4	0.0	99	10.1	0.0	99	10.8	0.0	4086	3628.3	18.4	4241	3624.2	15.3	5008	11.0	0.0
	20kroB100	69	3636.2	30.3	99	1362.2	0.0	99	12.7	0.0	83	3631.3	98.3	4668	3624.6	6.8	5008	13.0	0.0
	20kroC100	4	3633.3	96.0	94	3623.6	5.1	99	16.2	0.0	249	3641.9	95.0	3043	3622.8	39.2	5008	8.5	0.0
	20kroD100	5	3631.7	94.9	95	3632.2	4.0	99	10.1	0.0	3750	3624.9	25.1	4776	3635.7	4.6	5008	17.2	0.0
	20kroE100	97	3626.7	2.0	98	3621.1	1.0	99	7.9	0.0	325	3633.3	93.5	325	3628.2	93.5	5008	14.3	0.0
	20rat99	98	1511.3	0.0	98	166.3	0.0	98	7.8	0.0	5007	595.1	0.0	5007	360.5	0.0	5007	10.5	0.0
	20rd100	70	3633.4	29.3	99	53.9	0.0	99	6.7	0.0	5008	40.2	0.0	5008	121.4	0.0	5008	7.1	0.0
	21eil101	99	3622.5	1.0	100	1798.2	0.0	100	8.9	0.0	4831	3628.6	4.3	4933	3630.7	2.3	5050	12.4	0.0
	21lin105	104	3.8	0.0	104	4.7	0.0	104	6.8	0.0	5228	2185.6	0.0	5228	1737.9	0.0	5228	5.4	0.0
	22pr107	106	12.0	0.0	106	9.0	0.0	106	6.0	0.0	5363	64.4	0.0	5363	145.4	0.0	5363	5.6	0.0
	Average	**71.4**	**1755.4**	**16.9**	**87.5**	**953.2**	**0.7**	**88.1**	**7.2**	**0.0**	**3348.4**	**1813.9**	**22.3**	**3925.9**	**1671.8**	**10.8**	**4466.1**	**8.1**	**0.0**

6. Conclusions

The Set Orienteering Problem, where the tour of a single vehicle has to be calculated in order to collect the maximum possible profit from visiting clusters in the given available time, is the subject of the present report. We presented a new preprocessing rule, exploiting for the first time the limited available time, and a new constraint programming model to formally describe the problem. From an empirical point of view, the effectiveness of the new preprocessing rule is shown. Moreover, through solving the new constraint programming model with modern solvers, and therefore exploiting the high symmetry characterising the model, new state-of-the-art results for the instances commonly adopted in the literature for exact algorithms that improve on those of very recent publications are disclosed.

Author Contributions: Conceptualization, R.M. and D.H.S.; methodology, R.M.; software, R.M.; validation, R.M. and D.H.S.; formal analysis, R.M. and D.H.S.; investigation, R.M. and D.H.S.; resources, R.M.; data curation, R.M.; writing—original draft preparation, R.M. and D.H.S.; writing—review and editing, R.M. and D.H.S.; visualization, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available from [16].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Golden, B.; Levy, L.; Vohra, R. The orienteering problem. *Nav. Res. Logist.* **1987**, *3*, 307–318. [CrossRef]
2. Gunawan, A.; Lau, H.; Vansteenwegen, P. Orienteering problem: A survey of recent variants, solution approaches and applications. *Eur. J. Oper. Res.* **2016**, *2*, 315–332. [CrossRef]
3. Archetti, C.; Carrabs, F.; Cerulli, R. The set orienteering problem. *Eur. J. Oper. Res.* **2018**, *1*, 264–272. [CrossRef]
4. Montemanni, R.; Smith, D.H.; Gambardella, L.M. Ant colony systems for large sequential ordering problems. In Proceedings of the IEEE Swarm Intelligence Symposium, Honolulu, HI, USA, 1–5 April 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 60–67.
5. Dumez, D.; Lehuédé, F.; Péton, O. A large neighborhood search approach to the vehicle routing problem with delivery options. *Transp. Res. Part B* **2021**, *144*, 103–132. [CrossRef]
6. Dell'Amico, M.; Montemanni, R.; Novellani, S. Pickup and delivery with lockers. *Transp. Res. Part C* **2023**, *148*, 104022. [CrossRef]
7. Pěnička, R.; Faigl, J.; Saska, M. Variable neighborhood search for the set orienteering problem and its application to other orienteering problem variants. *Eur. J. Oper. Res.* **2019**, *3*, 816–825. [CrossRef]
8. Carrabs, F. A biased random-key genetic algorithm for the set orienteering problem. *Eur. J. Oper. Res.* **2021**, *3*, 830–854. [CrossRef]
9. Archetti, C.; Carrabs, F.; Cerulli, R.; Laureana, F. A new formulation and a branch-and-cut algorithm for the set orienteering problem. *Eur. J. Oper. Res.* **2024** , *314*, 446–465. [CrossRef]
10. Montemanni, R.; Dell'Amico, M. Solving the Parallel Drone Scheduling Traveling Salesman Problem via Constraint Programming. *Algorithms* **2023**, *16*, 40. [CrossRef]
11. Ramani, A.; Markov, I. Automatically Exploiting Symmetries in Constraint Programming. In Proceedings of the Conference on Recent Advances in Constraints, Lausanne, Switzerland, 23–25 June 2004; Lecture Notes in Computer Science; Volume 3419, pp. 60–67.
12. Gent, I.; Petrie, K.; Puget, J.F. Symmetry in Constraint Programming,. *Found. Artif. Intell.* **2006**, *2*, 329–376.
13. Walsh, T. General Symmetry Breaking Constraints. In Proceedings of the Conference on Principles and Practice of Constraint Programming, Nantes, France, 25–29 September 2006; Lecture Notes in Computer Science; Volume 4204.
14. Google. OR-Tools. Available online: https://developers.google.com/optimization/ (accessed on 6 February 2024).
15. Carrabs, F. (University of Salerno, Fisciano (SA), Italy). Personal communication, 2024.
16. Carrabs, F. The Set Orienteering Problem. Available online: https://github.com/fcarrabs/Set_Orienteering_Problem (accessed on 6 February 2024).