mathematics

Article

The Cost-Balanced Path Problem: A Mathematical Formulation and Complexity Analysis

Daniela Ambrosino *,† and Carmine Cerrone †

Department of Economics and Business Studies, University of Genoa, 16126 Genoa, Italy; carmine.cerrone@unige.it
* Correspondence: ambrosin@economia.unige.it
† These authors contributed equally to this work.

Abstract: This paper introduces a new variant of the Shortest Path Problem (SPP) called the Cost-Balanced Path Problem ($CBPP$). Various real problems can either be modeled as $BCPP$ or include $BCPP$ as a sub-problem. We prove several properties related to the complexity of the $CBPP$ problem. In particular, we demonstrate that the problem is NP-hard in its general version, but it becomes solvable in polynomial time in a specific family of instances. Moreover, a mathematical formulation of the $CBPP$, as a mixed-integer programming model, is proposed, and some additional constraints for modeling real requirements are given. This paper validates the proposed model and its extensions with experimental tests based on random instances. The analysis of the results of the computational experiments shows that the proposed model and its extension can be used to model many real applications. Obviously, due to the problem complexity, the main limitation of the proposed approach is related to the size of the instances. A heuristic solution approach should be required for larger-sized and more complex instances.

Keywords: shortest path problem; mixed-integer linear programming; cost-balanced paths

check for updates

Citation: Ambrosino, D.; Cerrone, C. The Cost-Balanced Path Problem: A Mathematical Formulation and Complexity Analysis. *Mathematics* **2022**, *10*, 804. https://doi.org/10.3390/math10050804

Academic Editor: Aleksandr Rakhmangulov

Received: 9 January 2022
Accepted: 1 March 2022
Published: 3 March 2022

1. Introduction

This paper introduces a new variant of the Shortest Path Problem (SPP) called the Cost-Balanced Path Problem ($CBPP$). The $CBPP$ is defined on a directed graph $G(N, A)$, where N is the set of nodes and A is the set of directed arcs. For each arc $(i, j) \in A$ is also defined a cost $c_{ij} \in \mathbb{R}$. Let the nodes $o, d \in N$, respectively, called origin and destination. A feasible solution of this problem is an acyclic path $p = ((o, n_i), (n_i, n_j), \ldots, (n_h, n_k), (n_k, d))$ in the graph G from node o to node d. Let $c(p) = \sum_{(i,j)\in p} c_{ij}$ the cost of the pah p, i.e., the sum of the cost of all the arcs used in the path p; the objective of $CBPP$ is the minimization of the absolute value of this cost: $MIN|c(p)|$.

Various real problems that present some elements in common with the cost-balanced path problem can either be modeled as $BCPP$ or include $BCPP$ as a sub-problem; some are here briefly discussed. The first mentioned problem is related to the path of an electric vehicle. The route choices of drivers of battery electric vehicles are affected by the many factors related to the battery recharge [1]. The cost-balanced path problem can be solved when defining the path that an electric vehicle has to perform for going from an origin point to a destination one, with the aim of maintaining the same level of electric charge. Suppose that a vehicle starts its trip in the origin node with a charge of 80% and has to arrive at a destination node with the same charge. During the trip, the vehicle can recharge the battery on the downhill roads, while the vehicle reduces its charge on the roads that go uphill. This problem can be formulated on a direct graph G where the weights of arcs represent the charge consumption (negative arc costs) and the recharge (positive costs). In this case, additional constraints are required, such as the level of electric charge to maintain along the whole path that can range from 0 to 100%.

In the bike-sharing systems [2], a particular problem linked to bikes management can be modeled as a $CBPP$. Suppose to have to re-locate bikes among a set of points that can be modeled as arcs in a direct graph G, with weights on the arcs representing the number of bikes to deliver (negative costs) and the number of bikes to pick up (positive costs). A vehicle has to perform a path in such a way to redistribute the bikes. In this example, the additional constraints are related to the number of bikes on the vehicle, that can range from zero to the vehicle capacity; moreover, additional requirements can be added for obtaining a constrained path concerning the duration, the number of arcs visited, the length, that, for example, should be maintained within a given range. Alternatively, suppose to have a depot and a vehicle that has to deliver a certain number of bikes to some points that are the locations for bikes. The vehicle has to deliver bikes to some locations, while eventually re-locate some bikes, that is, to pick up bikes from some locations. In the end, the vehicle has to finish its trip, possibly without bikes on board; the same additional requirements cited here above can be added.

This paper introduces a new problem, thus it can be helpful to summarize the novelties of this work in the following list: (i) definition of a new problem, the $CSPP$; (ii) proof that $CSPP$ is NP-hard; (iii) proof that it is possible to solve the $CSPP$ in polynomial time under specific configurations of the arc costs; (vi) first mathematical formulation for $CSPP$.

The remaining of the paper is organized as follows. Section 2 summarizes the literature related to $BCPP$. Section 3 presents an evaluation of the $BCPP$ complexity, the $BCPP$ mathematical formulation and some model extensions, while Section 4 reports the computational experiments for the validation of the proposed model and its extensions. Section 5 gives some conclusions and perspectives.

2. Literature Review

To the authors' knowledge, the $BCPP$ has never been studied in the literature, although there are many variants of the classic SPP, and there is a paper related to the Traveling Salesman Problem (TSP) that introduces the same objective function of $BCPP$ [3]. The authors of [3] introduce the cost-balanced TSP, in which the main objective is to find a Hamiltonian cycle with total travel cost as close as possible to 0. The authors assumed a cost/length matrix, while negative costs are allowed. To solve the cost-balanced TSP, they proposed a variable neighborhood search algorithm. A similar problem is the balanced TSP, which is related to the uniform (equitable) distribution of resources [4]. In [5], the multiple balanced traveling salesmen problem is proposed to model and optimize the problems with multiple objectives (salesmen). The goal is to find m Hamiltonian cycles in G by minimizing the difference between the highest edge cost and the smallest edge cost in the tours. The SPP [6,7] and many variants have been proposed in the literature for facing problems arising in various fields, together with ever more efficient algorithms (see, for example, in [8–11]). Although the SPP can be solved in polynomial time using various algorithms, many of its variants are known to be NP-hard. Among these variants of the SPP, in the k-Color Shortest Path Problem proposed by Ferone et al. [12,13], the classic SPP is solved on graphs with colored arcs. In the recent Steiner bi-objective Shortest Path Problem introduced in [14], the authors present this new variant of the SPP capable of preprocessing data to solve the well-known vehicle routing problem. SPP in which the cost of the arcs is not known in advance has been studied in the recent literature [15,16]. Stochastic shortest path (SSP) dealing with applications in routing problems and in road networks can be found in [17,18]. Another problem on graphs linked to the balance concept is the balanced trees [19], which are the appropriate structures (balanced tree structures) for managing networks with the aim of balancing two objectives. The constrained path has been studied in [20]; the authors proposed a robust formulation for the Resource-Constrained Shortest Path Problem that is the problem of determining a path p from an origin to a destination with the smallest cost, such that the consumption of a given resource for that path is lower or equal to the maximum amount of available resource.

In the following section the complexity of *CSPP* is investigated and a mathematical formulation is proposed.

3. Problem Complexity and Mathematical Formulation for *CBPP*

Many variants of *SPP* are known to be NP-hard; thus, in this section, before presenting a mathematical formulation for modeling and solving *CBPP*, the problem complexity is investigated. In particular, thanks to a reduction algorithm [21,22], it has been proved that the problem is NP-hard in its general form.

3.1. Problem Complexity

Theorem 1. *If not all costs have the same sign, the Cost-Balanced Path Problem is NP-hard.*

Proof of Theorem 1. To prove the theorem, we will describe a reduction algorithm which in polynomial time, reduces the classic Hamiltonian Path Problem (*HPP*) in an instance of the Cost-Balanced Path Problem. The *HPP* is a classic problem belonging to the NP-complete complexity class [23]. Let a directed graph $G(N, A)$ where N is a set of nodes, and A is a set of arcs. Let us suppose we want to compute the Hamiltonian path that goes from node h to node k of the graph G. We create the graph $G'(N', A')$ such that $N' = N \cup \{o\}$, $A' = A \cup \{(o, h)\}$. We create the cost c_{ij} such that $c_{ij} = 1\ \forall (ij) \in A$ and $c_{oh} = 1 - |N|$. Considering that $|N'| = |N| + 1$ and that the longest path in G' can contain $|N|$ arcs. Considering that in every feasible solution the arc (o, h) will always be present and that the value of a solution with k arcs will be equal to $1 - |N| + (k - 1)$. If the value of the solution of *CBPP* on the graph G' is equal to zero, then in G, there is a Hamiltonian path between the nodes h and k (see Figure 1). □

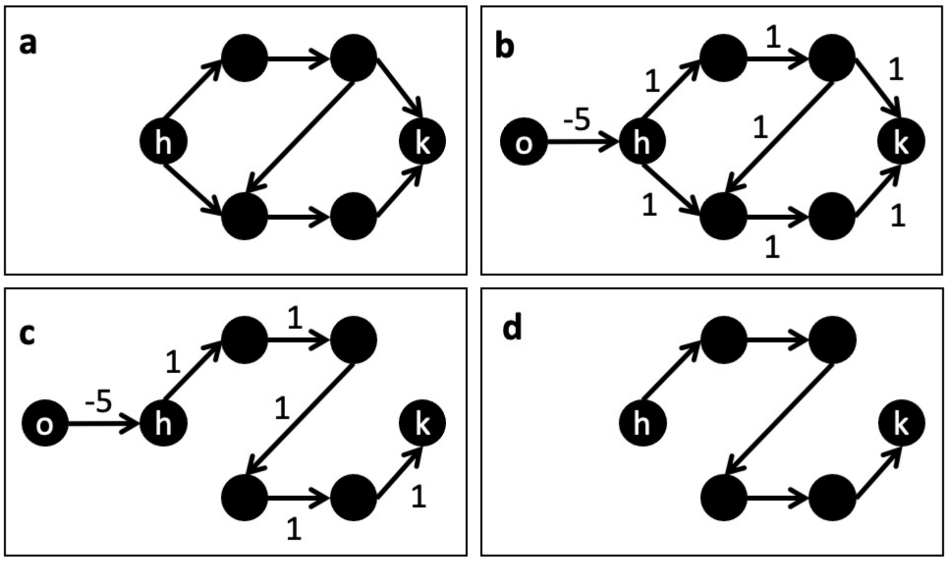

Figure 1. (**a**) An example of a graph G. (**b**) Graph G' derived from G. (**c**) A solution of *CBPP* with cost zero. (**d**) The Hamiltonian path.

Figure 1 shows an example useful to understand Proof of Theorem 1. In Figure 1a, a direct graph (i.e., G) with six nodes is depicted, while in Figure 1b, a graph G' derived from G is represented: node o has been added together with the weights for the arcs. In Figure 1c, the solution for the *CBPP* is shown. Finally, the Hamiltonian path from node h to node k, is depicted in Figure 1d).

Corollary 1 (Corollary of Theorem 1)**.** *Given a generic instance of CBPP, if even one cost has the opposite sign to the others, then the problem is NP-hard.*

Proposition 1. *If all costs are non-negative $c_{ij} \geq 0$, the problem is equivalent to the classic Shortest Path Problem, therefore it can be solved in polynomial time.*

Proposition 2. *If all costs are non-positive $c_{ij} \leq 0$, the problem is equivalent to the classic Shortest Path Problem, therefore it can be solved in polynomial time.*

Proof of Proposition 2. By inverting the sign of each cost, we will obtain a scenario in which all costs are positive. Proposition 1 assures us that we can solve the resulting problem in polynomial time. The obtained solution is an optimal solution also for the initial problem with the unique difference that the value of the objecting function is negative. □

A graph with particular characteristics for *CBPP* is the graph in which the cost of the arc is a function of the elevation difference of the two nodes associated with the arc. This graph could represent points positioned at different altitudes (see Figure 2).

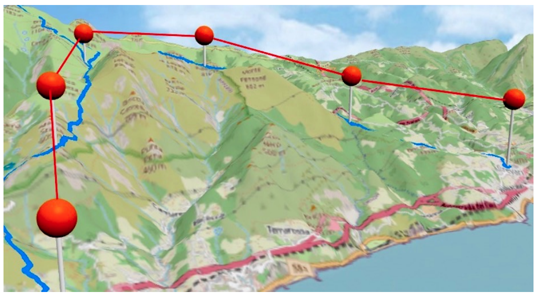

Figure 2. Example of an altimetric graph.

Proposition 3 (Elevation difference). *Given a directed graph $G(N, A)$ in which for each node $i \in N$ is defined a value $v_i \in \mathbb{R}$, and such that $c_{ij} = v_j - v_i \; \forall (ij) \in A$. The Cost-Balanced Path Problem is solvable in polynomial times.*

Proof of Proposition 3. Given a graph G created as described in Proposition 3, let be p a path in G, such that $p = ((n_1, n_2), (n_2, n_3), (n_3, n_4), \ldots, (n_h, n_k))$, $c(p) = c_{12} + c_{23} + c_{34} + \ldots + c_{hk} = (v_2 - v_1) + (v_3 - v_2) + (v_4 - v_3) + \ldots. + (v_k - v_h) = -v_1 + (v_2 - v_2) + (v_3 - v_3) + (v_4 - v_4) + \ldots + (v_h - v_h) + v_k \implies c(p) = v_k - v_1$. This implies that the cost of a path depends only on the starting node and the destination node, so each path is also optimal (see Figure 3). □

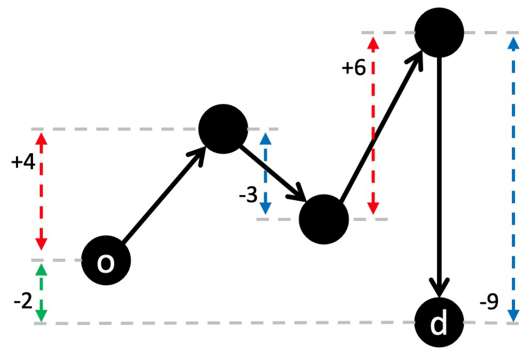

Figure 3. Example of a graph in which the cost of an arc depends on the elevation of the nodes.

3.2. Mathematical Model

In this sub-section, a model for solving *CBPP* is presented; it is a mixed integer linear programming with binary variables (MILP). Let us introduce the following decision variables:

$x_{ij} \in \{0, 1\}, \forall i, j \in N$: $x_{ij} = 1$ if and only if arc (i, j) is included in the problem solution.

$t_i, \forall i \in N$ that represents the flow leaving node i and is used to prevent the creation of loops in the solution.

b represents the costs (in absolute value) of the optimal path.

The resulting model is the following:

$$z = Min\ b \tag{1}$$

subject to:

$$\sum_{(i,j)\in A} c_{ij}x_{ij} \le b \tag{2}$$

$$-\sum_{(i,j)\in A} c_{ij}x_{ij} \le b \tag{3}$$

$$\sum_{(o,j)\in A} x_{oj} = 1 \tag{4}$$

$$\sum_{(i,d)\in A} x_{id} = 1 \tag{5}$$

$$\sum_{(j,i)\in A} x_{ji} - \sum_{(i,j)\in A} x_{ij} = 0 \qquad \forall i \in N \setminus \{o,d\} \tag{6}$$

$$\sum_{(i,j)\in A} x_{ij} \le 1 \qquad \forall j \in N \setminus \{o,d\} \tag{7}$$

$$\sum_{(i,j)\in A} x_{ij} \le 1 \qquad \forall i \in N \setminus \{o,d\} \tag{8}$$

$$t_o = 0 \tag{9}$$

$$t_j - t_i \ge 1 - |N|(1 - x_{ij}) \qquad \forall (i,j) \in A \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{11}$$

$$t_i \in [0, |N| - 1] \qquad \forall i \in N \tag{12}$$

Equation (1) minimizes variable b that represents the cost of the selected path in absolute value. Variable b is defined thanks to Equations (2) and (3). Equations (4) and (5) impose that one arc leaves the origin node o, and one arc enters the destination node d. (6) impose, for each node of the network that is different from either the origin or the destination node, that the number of arcs entering the node is equal to the number of arcs exiting it. Equations (7) and (8) impose that at most one arc can enter in and exit from each node, except for the origin and the destination ones. Equations (9) and (10) defines variables t_i; t_o is set to zero (i.e., from the origin node the outflow is equal to zero), while t_j is set greater than the flow leaving node i, if arc (i,j) is selected. Finally, in (11) and (12) the decision variables are defined.

3.3. Model Extensions

In the introduction, some real applications that can be solved by the here above proposed model have been briefly described. Unfortunately, some additional constraints should be required, and thus in this sub-section, some of the additional constraints for model (1)–(12) are described.

$CBPP$ has the objective of cost balancing instead of cost minimization. The cost that it is necessary to balance may represent a measure of a level of a particular element that has to be maintained near a pre-defined value (for example, the electric charge, the load of a vehicle, etc.). Each decision, expressed in the graph by the selection of an arc, may increase or decrease the level of the considered element. The scope is to take a sequence of decisions in such a way to have at the end of the process the same starting level that, in particular cases, can be zero.

In real applications, there is often the necessity to maintain this level within given upper and lower limits after each decision, that is, along the selected path. This means that, i.e., in the example of the electric car, the charge must always be within a lower and an upper bound. The same can be required for the cargo loaded on a vehicle.

Constraints for limiting the variance of the level within the given interval (a^{min}, a^{max}) are based on flow variables defined as follows:

$f_{ij}, \forall i, j \in N$: f_{ij} represents the level reached at node i, that will leave node i for reaching node j if and only if arc (i, j) is included in the selected path, i.e., $x_{ji} = 1$.

Thanks to Equations (13) and (14), the flow (f_{ij}) on each selected arc must be less or equal to its maximum value and greater or equal to the minimum required, while thanks to Equation (15) the outflow from the origin node (f_{oj}) is fixed equal to the starting level (a_o).

Equation (16) gives the flow conservation constraints. For each node i, different from either the origin node or the destination one, the flow that leaves node i is equal to the flow that leaves the initial node of the arc entering in i, plus the cost of arc entering in node i.

$$f_{ij} \leq a^{max} x_{ij}, \forall (i, j) \in A \tag{13}$$

$$f_{ij} \geq a^{min} x_{ij}, \forall (i, j) \in A \tag{14}$$

$$\sum_{(d,j) \in A} f_{dj} = a_o \tag{15}$$

$$\sum_{(j,i) \in A} f_{ji} + \sum_{(j,i) \in A} c_{ji} x_{ji} = \sum_{(i,j) \in A} f_{ij}, \forall i \in N \setminus \{o, d\} \tag{16}$$

Thanks to the above constraints, we are able to balance the cost and maintain it in the given required interval along the whole path.

Sometimes, together with the aim of cost balancing some other objectives must be included in the problem. In fact, when dealing with paths, the most common problem is the shortest path problem. For example, in the problem of the electric car, it should be required to have a path not too expensive in terms of either kilometers traveled or times. In this case, it is possible to insert an additional constraint that permits to find a path from origin to destination no longer than a given % of the shortest path.

Let d_{ij} be the distance associated to the arc (i, j), and c^{SP} the distance associated to the shortest path from the origin node to the destination one in the graph under investigation, α the percentage of deterioration accepted, the resulting constraint is the following:

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \leq (1 + \alpha) c^{SP} \tag{17}$$

In other cases, the limitations may concern the length of the path in terms of number of arcs belonging to the path; the model can be extended by simply adding the following constraints that are related, respectively, to the maximum number of arcs that can build the path (b^{max}) and the minimum number of arcs to select (b^{min}).

$$\sum_{(i,j) \in A} x_{ij} \leq b^{max} \tag{18}$$

$$\sum_{(i,j) \in A} x_{ij} \geq b^{min} \tag{19}$$

4. Results

In this section, the computational results obtained by applying the proposed mathematical model (1)–(12) are described. Some computational experiments related to the extended model are presented too. The computational campaigns are based on some generated instances described in the following subsection.

The $MILP$ model has been implemented in Java, using CPLEX version 12.8 as a solver. The computational tests were performed on a MacBook Pro, with a 2.9 GHz Intel i9 processor and 32 GB of RAM. Figure 4 shows the flow chart of the proposed approach. Step 1 load the input graph $G(N, A)$ from the text file. Step 2 creates the mathematical model. Its complexity depends on the number of constraints created, which is in the order of $|A|$. Using the $MILP$ solver of the $CPLEX$ software, in Step 3, the problem is solved. The computation time for Step 3 is exponential [24] as stated in Theorem 1. In Step 4, all the values associated with the model decision variables are extracted to create a textual representation of the solution.

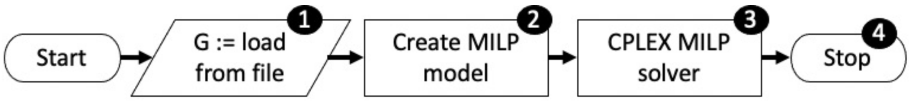

Figure 4. Algorithm flowchart.

4.1. Instances

The proposed mathematical model has been validated with two sets of generated instances for the above described problem. The first set of instances, named *Grid*, is characterized by complete square grids where each node is connected to its four neighbors. In the name of these instances, the first value represents the number of nodes; the second value represents the size of the grid. The second set of instances named *Rand* is characterized by connected graphs in which each node is randomly connected to other nodes until the desired density is reached. In the name of these instances, the first value represents the number of nodes; the second value represents the percentage of arcs incident on each vertex. The costs associated with the arcs of each instance were generated following 5 different schemes.

[−10, 10] Random homogeneous distribution of costs in the range $[-10, 10]$.
[−100, 100] Random homogeneous distribution of costs in the range $[-100, 100]$.
[−1k, 1k] Random homogeneous distribution of costs in the range $[-1000, 1000]$.
EL After associating a random height to each node, the cost of the arc represents the displacement in height.
P-EL Perturbation of the 1% random of the EL costs.

4.2. Computational Results for the Proposed Model

Tables 1–3 show the results of the computational tests performed on the grid instances. Each row reports the average of five instances. The last row of each table AVG is the average of all solved scenarios. We used 1800 s as the time limit for the CPLEX solver (Step 3). This implies that in the event of a higher running time, the optimality of the obtained solution is not guaranteed. Table 1 shows the computational times. It is interesting to note that the running time is mainly related to the costs associated with the arcs rather than to the size of the graph. Table 2 shows the number of zero-value solutions identified. For instances with random costs, in the scenarios with homogeneous weight distribution ($[-10, 10]$, $[-100, 100]$, $[-1k, 1k]$), it is always possible to obtain a solution with a cost equal to zero. Using the EL policy to create the costs, all paths will have a cost equal to the difference in height between the source node and the destination node. It is interesting to see that analyzing the P-EL policy is sufficient a 1% perturbation of the EL policy cost in order to identify solutions with a cost equal to zero, in particular as the graph size increases. Table 3 shows the obtained objective function values. The analysis of Table 3 shows that as the size of the graph increases, the solution for the P-EL policy will tend to approach zero.

Table 1. Running times for the grid instances.

Instance	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
Grid_100_10	0.27	0.79	1.71	0.02	38.23
Grid_225_15	0.43	0.95	3.60	0.07	1502.42
Grid_400_20	0.69	1.66	4.30	0.11	961.81
AVG	**0.46**	**1.13**	**3.20**	**0.07**	**834.15**

Table 2. Number of solutions with an objective function value equal to zero.

Instance	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
Grid_100_10	5	5	5	0	0
Grid_225_15	5	5	5	0	0
Grid_400_20	5	5	5	0	3
AVG	**5**	**5**	**5**	**0**	**1**

Table 3. Objective function value for the grid instances.

Instance	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
Grid_100_10	0	0	0	4467	3806
Grid_225_15	0	0	0	4809	3405
Grid_400_20	0	0	0	2920	878
AVG	**0**	**0**	**0**	**4065**	**2696**

Tables 4–6 show the results of the computational tests performed on the random generated instances. As before, each row reports the average of five solved instances, while the two rows AVG refer to the average of all solved scenarios, respectively, for the instances with 100 and 200 nodes. Table 4 shows the computational times. In this test, the running time remains highly dependent on the used cost scheme, but the P-EL scheme is much easier to solve than the [−1k, 1k] scheme. Table 5 shows the number of zero-value solutions identified. For instances with random costs, in the scenarios with homogeneous weight distribution ([−10, 10],[−100, 100],[−1k, 1k]), it is always possible to obtain a solution with a cost equal to zero. This test confirms the results obtained previously for the grid instances. In this scenario, it becomes even more evident that the cost scheme P-EL tends as the graph grows to produce instances with cost zero solution (see also Table 6). Table 6 shows the obtained objective function values. Considering that the execution of the model stops reaching a solution equal to zero (Lower Bound), we justify the computational times shown in Table 4.

Table 4. Running times for the random instances.

Instance	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
Rand_100_02	0.06	0.09	1.28	0.01	0.05
Rand_100_03	0.13	0.29	1.17	0.02	0.13
Rand_100_04	0.18	0.29	2.40	0.02	0.82
Rand_100_05	0.12	0.48	2.37	0.03	0.50
Rand_100_10	0.13	0.42	2.25	0.05	0.23
Rand_100_20	0.18	0.68	4.14	0.10	0.46
AVG	**0.13**	**0.38**	**2.27**	**0.04**	**0.36**
Rand_200_02	0.30	1.18	3.91	0.05	0.50
Rand_200_03	0.22	1.11	4.53	0.06	0.71
Rand_200_04	0.16	0.35	6.63	0.09	1.30
Rand_200_05	0.25	0.59	6.28	0.11	0.66
Rand_200_10	0.23	3.88	4.95	0.18	1.01
Rand_200_20	0.15	12.66	23.08	0.38	6.66
AVG	**0.22**	**3.30**	**8.23**	**0.14**	**1.81**

Table 5. Number of solutions with an objective function value equal to zero.

Instance	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
Rand_100_02	5	5	5	0	1
Rand_100_03	5	5	5	0	0
Rand_100_04	5	5	5	0	2
Rand_100_05	5	5	5	0	1
Rand_100_10	5	5	5	0	2
Rand_100_20	5	5	5	0	2
AVG	**5**	**5**	**5**	**0**	**1**
Rand_200_02	5	5	5	0	1
Rand_200_03	5	5	5	0	1
Rand_200_04	5	5	5	0	1
Rand_200_05	5	5	5	0	2
Rand_200_10	5	5	5	0	2
Rand_200_20	5	5	5	0	4
AVG	**5**	**5**	**5**	**0**	**2**

Table 6. Objective function value for the random instances.

Instance	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
Rand_100_02	0	0	0	3634	1435
Rand_100_03	0	0	0	3040	6005
Rand_100_04	0	0	0	3358	434
Rand_100_05	0	0	0	2343	2982
Rand_100_10	0	0	0	3330	1888
Rand_100_20	0	0	0	2107	1762
AVG	**0**	**0**	**0**	**2969**	**2418**
Rand_200_02	0	0	0	2959	2835
Rand_200_03	0	0	0	4109	2559
Rand_200_04	0	0	0	4240	1545
Rand_200_05	0	0	0	2550	1762
Rand_200_10	0	0	0	2512	1676
Rand_200_20	0	0	0	2721	74
AVG	**0**	**0**	**0**	**3182**	**1742**

4.3. Results for the Extended Model

In this section, some results related to the extended model are presented. In particular, these tests are based on the 50 instances named Grid_100_10 and Grid_225_15. In all experiments, a^{max} is equals to $-a^{min}$. Looking at Table 7, it is possible to note that by decreasing the value of $alpha$, the computational time decreases, according to the decrease in the dimension of the admissible region. On the other hand, by introducing in the model the constraints associated with the parameter a^{max}, the computational time increases, according to the increase in the number of decision variables and constraints associated with the problem. As in Table 8, we can see that obviously, as the number of constraints increases, it becomes increasingly challenging to identify zero-sum solutions. Table 9 shows that as the number of constraints increases, it becomes even more difficult to identify feasible solutions: the values reported in round brackets indicate the number of unfeasible solutions. Analyzing Tables 8 and 9, it is possible to state that as the size of the graph increases, the quality of the solutions worsens less by adding further constraints. This is probably due to the increase in alternative paths between source and destination nodes.

Table 7. Running times in seconds.

$\|N\|$	α	a^{max}	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
100	∞	∞	0.27	0.79	1.71	0.02	38.23
	0.2	∞	0.14	0.54	1.43	0.03	0.04
	0.1	∞	0.13	0.27	0.29	0.03	0.04
	0.1	4C	0.27	0.77	1.12	0.15	0.16
	0.1	3C	0.39	0.60	0.88	0.14	0.16
	0.2	3C	0.73	0.86	2.38	0.22	0.20
225	∞	∞	0.43	0.95	3.60	0.07	1502.42
	0.2	∞	0.99	10.15	113.64	0.13	0.50
	0.1	∞	1.47	5.09	11.57	0.16	0.24
	0.1	4C	4.94	30.00	30.21	0.75	0.49
	0.1	3C	5.81	9.16	12.77	0.90	0.79
	0.2	3C	13.15	19.65	144.18	0.85	0.60

Table 8. Number of solutions with an objective function value equal to zero.

$\|N\|$	α	a^{max}	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
100	∞	∞	5	5	5	0	0
	0.2	∞	3	2	1	0	0
	0.1	∞	2	0	1	0	0
	0.1	4C	2	0	0	0	0
	0.1	3C	2	0	0	0	0
	0.2	3C	2	2	0	0	0
225	∞	∞	5	5	5	0	0
	0.2	∞	5	4	2	0	0
	0.1	∞	4	2	0	0	0
	0.1	4C	2	1	0	0	0
	0.1	3C	1	0	0	0	0
	0.2	3C	4	2	1	0	0

Table 9. Objective function value—values in round brackets indicate the number of unfeasible solutions.

$\|N\|$	α	a^{max}	[−10, 10]	[−100, 100]	[−1k, 1k]	EL	P-EL
100	∞	∞	0	0	0	4467	3806
	0.2	∞	6	15	13	4467	4379
	0.1	∞	12	95	233	4467	4402
	0.1	4C	3 (2)	15 (2)	260	4467	4402
	0.1	3C	3 (2)	15 (2)	179 (1)	4467 (1)	5003 (1)
	0.2	3C	3 (1)	4 (2)	100	4467	4379
225	∞	∞	0	0	0	4809	3405
	0.2	∞	0	1	10	4809	4870
	0.1	∞	1	4	175	4809	4900
	0.1	4C	3	5 (1)	5 (2)	4809	4900
	0.1	3C	1 (2)	17 (2)	18 (3)	4809	4900
	0.2	3C	1 (1)	26	35	4809	4870

5. Conclusions

This paper deals with the Cost-Balanced Path Problem ($CBPP$), a variant of the classic Shortest Path Problem introduced in this paper for the first time. The characteristic of this problem is that it can be used as a sub-problem to model many real scenarios. Using the mixed-integer linear programming model introduced in Section 3.2, we computed the optimal solution for many test instances. It is interesting to note that analyzing the results shown in Section 4, in the case of uniform distribution of the costs of the arcs, there is always an optimal solution with an objective function value equal to zero. To prevent or make the presence of solutions with an objective function value equal to zero rarer, smart methods for defining the cost of the edges (EL, $P - EL$) have been used. Note that when the model reaches an objective function value equal to zero, it stops instantaneously having reached its lower bound; this implies that the computational time for instances that do not have zero as an optimal solution is significantly higher. Considering these observations, the future developments for this work are manifold. First of all, it would be interesting to develop instance generators capable of preventing or minimizing the presence of zero solutions in order to create computationally complex instances. Using more complex instances, realizing heuristic or meta-heuristic approaches for this problem would become necessary. A constructive approach based on the Dijkstra algorithm [25] improved through the Carousel Greedy, an enhanced Greedy algorithm proposed in [26,27], might identify a feasible solution to the problem. According to the authors' experience, the tabu search, a technique introduced by Glover [28] and widely used in the literature, also by the authors of this work, for example, in [29], might be used to improve the Greedy solution.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request.

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

Abbreviations

The following abbreviations are used in this manuscript:

SPP	Shortest Path Problem
CBPP	Cost-Balanced Path Problem
HPP	Hamiltonian Path Problem
MILP	Mixed Integer Linear Programming

References

1. He, F.; Yin, Y.; Lawphongpanich, S. Network equilibrium models with battery electric vehicles. *Transp. Res. Part B Methodol.* **2014**, *67*, 306–319. [CrossRef]
2. Dell'Amico, M.; Hadjicostantinou, E.; Iori, M.; Novellani, S. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega* **2014**, *45*, 7–19. [CrossRef]
3. Akbay, M.; Kalayci, C. A Variable Neighborhood Search Algorithm for Cost-Balanced Travelling Salesman Problem. In *Metaheuristics Summer School*; Springer: Cham, Switzerland, 2018; pp. 23–36.
4. Larusic, J.; Punnen, A.P. The balanced traveling salesman problem. *Comput. Oper. Res.* **2011**, *38*, 868–875. [CrossRef]
5. Dong, X.; Xu, M.; Lin, Q.; Han, S.; Li, Q.; Guo, Q. IT algorithm with local search for large scale multiple balanced traveling salesmen problem. *Knowl.-Based Syst.* **2021**, *229*, 107330. [CrossRef]
6. Gallo, G.; Pallotino, S. Shortest path algorithms. *Ann. Oper. Res.* **1988**, *13*, 3–79. [CrossRef]
7. Cherkassy, B.V.; Goldberg, A.V.; Radzik, T. Shortest path algorithms: Theory and experimental evaluation. *Math. Program.* **1996**, *73*, 129–174. [CrossRef]
8. Fu, L.; Sun, D.; Rilett, L.R. Heuristic shortest path algorithms for transportation applications: State of the art. *Comput. Oper. Res.* **2006**, *33*, 3324–3343. [CrossRef]
9. Raith, A.; Ehrgott, M. A comparison of solution strategies for bi-objective shortest path problems. *Comput. Oper. Res.* **2009**, *36*, 1299–1331. [CrossRef]
10. Panda, M.; Mishra, A. A survey of shortest-path algorithms. *Int. J. Appl. Eng. Res.* **2018**, *13*, 6817–6820.
11. Yuan, H.; Hu, J.; Song, Y.; Li, Y.; Du, J. A new exact algorithm for the shortest path problem: An optimized shortest distance matrix. *Comput. Ind. Eng.* **2021**, *158*, 107407. [CrossRef]
12. Ferone, D.; Festa, P.; Pastore, T. The k-color shortest path problem. In *Advances in Optimization and Decision Science for Society, Services and Enterprises*; Springer: Cham, Switzerland, 2019; pp. 367–376.
13. Ferone, D.; Festa, P.; Fugaro, S.; Pastore, T. A dynamic programming algorithm for solving the k-Color Shortest Path Problem. *Optim. Lett.* **2021**, *15*, 1973–1992. [CrossRef]

14. Ticha, H.B.; Absi, N.; Feillet, D.; Quilliot, A. The Steiner bi-objective shortest path problem. *EURO J. Comput. Optim.* **2021**, *9*. [CrossRef]
15. Ketkov, S.S.; Prokopyev, O.A.; Burashnikov, E.P. An approach to the distributionally robust shortest path problem. *Comput. Oper. Res.* **2021**, *130*, 1105212. [CrossRef]
16. Zhang, D.; Wallace, S.W.; Guo, Z.; Dong, Y.; Kaut, M. On scenario construction for stochastic shortest path problems in real road networks. *Transp. Res. Part E Logist. Transp. Rev.* **2021**, *152*, 102410. [CrossRef]
17. Ehmke, J.F.; Campbell, A.M.; Thomas, B.W. Data-driven approaches for emissions-minimized paths in urban areas. *Comput. Oper. Res.* **2016**, *67*, 34–47. [CrossRef]
18. Prakash, A.A. Pruning algorithm for the least expected travel time path on stochastic and time-dependent networks. *Transp. Res. Part B Methodol.* **2018**, *108*, 127–147. [CrossRef]
19. Moharam, R.; Morsy, E. Genetic algorithms to balanced tree structures in graphs. *Swarm Evol. Comput.* **2017**, *32*, 132–139. [CrossRef]
20. Di Puglia Pugliese, L.; Guerriero, F.; Poss, M. The Resource Constrained Shortest Path Problem with uncertain data: A robust formulation and optimal solution approach. *Comput. Oper. Res.* **2019**, *107*, 140–155. [CrossRef]
21. Carrabs, F.; Cerrone, C.; Cerulli, R.; Silvestri, S. On the complexity of rainbow spanning forest problem. *Optim. Lett.* **2018**, *12*, 443–454. [CrossRef]
22. Carrabs, F.; Cerrone, C.; Cerulli, R.; Silvestri, S. The rainbow spanning forest problem. *Soft Comput.* **2018**, *22*, 2765–2776. [CrossRef]
23. Garey, M.; Johnson, D. *Computers and Intractability*, 3rd ed.; Freeman: San Francisco, CA, USA, 1979.
24. Schrijver, A. *Theory of Linear and Integer Programming*; John Wiley & Sons: Hoboken, NJ, USA, 1998.
25. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
26. Cerrone, C.; Cerulli, R.; Golden, B. Carousel greedy: A generalized greedy algorithm with applications in optimization. *Comput. Oper. Res.* **2017**, *85*, 97–112. [CrossRef]
27. Carrabs, F.; Cerrone, C.; D'Ambrosio, C.; Raiconi, A. Column generation embedding carousel greedy for the maximum network lifetime problem with interference constraints. In *International Conference on Optimization and Decision Science*; Springer: Cham, Switzerland, 2017; pp. 151–159.

28. Glover, F. Tabu search—Part I. *ORSA J. Comput.* **1989**, *1*, 190–206. [CrossRef]
29. Carrabs, F.; Cerrone, C.; Cerulli, R. A tabu search approach for the circle packing problem. In Proceedings of the 17th International Conference on Network-Based Information Systems, Salerno, Italy, 10–12 September 2014; pp. 165–171.