electronics

MDPI

Article

Trust-Based Recommendation for Shared Mobility Systems Based on a Discrete Self-Adaptive Neighborhood Search Differential Evolution Algorithm

Fu-Shiung Hsieh

Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan; fshsieh@cyut.edu.tw

Abstract: Safety is one concern that hinders the acceptance of ridesharing in the general public. Several studies have been conducted on the trust issue in recent years to relieve this concern. The introduction of trust in ridesharing systems provides a pragmatic approach to solving this problem. In this study, we will develop a trust-aware ridesharing recommender system decision model to generate recommendations for drivers and passengers. The requirements of trust for both sides, drivers and passengers, are taken into consideration in the decision model proposed in this paper. The decision model considers the factors in typical ridesharing systems, including vehicle capacities, timing, location and trust requirements, etc. The decision model aims to determine the shared rides that minimize cost while respecting the trust and relevant constraints. As the decision problem is a nonlinear integer programming problem, we combine a self-adaptive neighborhood search with Differential Evolution to develop an algorithm to solve it. To assess the effectiveness of the proposed algorithm, several other evolutionary computation approaches are also applied to solve the same problem. The effectiveness assessment is done based on the performance of applying different algorithms to find solutions for test cases, to provide a guideline for selecting a proper solution approach.

Keywords: shared mobility; trust; ridesharing; evolutionary computation

Citation: Hsieh, F.-S. Trust-Based Recommendation for Shared Mobility Systems Based on a Discrete Self-Adaptive Neighborhood Search Differential Evolution Algorithm. *Electronics* **2022**, *11*, 776. https://doi.org/10.3390/electronics11050776

Academic Editor: George Angelos Papadopoulos

Received: 28 January 2022
Accepted: 28 February 2022
Published: 2 March 2022

1. Introduction

Shared mobility refers to transportation services and resources shared among users. Depending on the resources used, several types of transportation modes emerge, e.g., bike-sharing, scooter sharing, carsharing and ridesharing. With shared mobility, the number of e-hailing trips roughly tripled in four years. It has created over 40 million e-hailing trips daily on major e-hailing platforms [1]. Due to the potential benefits to reduce overall costs, energy consumption and greenhouse gas emissions, shared mobility sparks many new research directions and opportunities. Shared mobility may take different forms in the shared economy era. Ridesharing and carpooling are two well-known transportation modes in shared mobility. In Ref. [2,3], an early survey of ridesharing problems was discussed. Ridesharing services rely on an effective recommender system to generate recommendations for users. A ridesharing recommender system must take into account the factors of location, time and cost savings. The research issues relevant to the design of such ridesharing recommender systems include modeling, optimization and allocation of cost savings. These research issues pose challenges in the development of ridesharing recommender systems [4,5].

Despite the numerous potential advantages of ridesharing, e.g., reducing energy consumption, travel cost, greenhouse gas emissions and providing a flexible alternative other than public transport with fixed routing, the concern about safety and trust still hinders the progress for accepting the ridesharing transportation mode. Safety and trust are the primary concern for not adopting ridesharing. One approach to ensuring safety and trust

is to exploit the social relation information about potential ridesharing participants from social media [6]. Therefore, several studies on ensuring safety/trust in ridesharing through social networks have been carried out in recent years [7–12]. The decision models to generate recommendations for the trust-based ridesharing problem are generally formulated as non-linear integer programming problems, which consist of non-linear constraints and discrete decision variables. These problems are typically non-convex and the computational complexity grows with the scale of the problems. Therefore, an exact optimization method cannot be applied. Instead, approximate methods, such as evolutionary computation approaches, are adopted to find solutions. For example, ridesharing systems that consider the trust factor have been studied in [13–15], based on Particle Swarm Optimization (PSO) [16] and Differential Evolution (DE) [17] approaches.

In the literature on evolutionary computation, PSO [16] and DE [17] have over two decades of history and have been successfully applied to problems in several domains. Many variants of Particle Swarm Optimization approaches and Differential Evolution approaches have been proposed. For example, a discrete version of a Particle Swarm Optimization algorithm has been applied to solve to ridesharing problem with trust requirements in [13]. In Ref. [14], a discrete version of Differential Evolution has been developed to solve the trust-based ridesharing problem. Combining a neighborhood search with a Differential Evolution approach is adopted in [15] to solve the trust-based ridesharing problem. However, the effectiveness of applying different variants of DE approaches to solve the trust-based ridesharing decision problem needs to be studied. This study aims to develop a variant of a metaheuristic algorithm to solve the trust-based ridesharing decision problem with the goal to find a more effective solution algorithm. To achieve this goal, we will combine a self-adaptive neighborhood search with Differential Evolution (SaNSDE) to develop an algorithm. In particular, we will study the influence of the learning period parameter on the efficiency of the proposed discrete SaNSDE algorithm. We will assess the effectiveness of the proposed algorithm by comparing the proposed algorithm with other approaches, based on the results obtained by applying these algorithms to solve several test cases of the trust-based ridesharing problems.

The decision problem studied in this paper is different from the studies of [18,19], in that trust requirements are considered. This paper is also different from our recent study on the allocation of cost savings in [20], where trust requirements are not considered. The trust model adopted in this paper is different from the ones used in [9,11]. The contributions of this paper include a problem formulation for trust-based ridesharing systems and the development of a more effective self-adaptive neighborhood search algorithm for solving the trust-based ridesharing problem. The remainder of this paper is structured as follows. In Section 2, we first give a review of the literature relevant to the enhancement of trust in ridesharing systems. We will present the decision model of trust-based ridesharing systems in Section 3. We will propose several algorithms based on several variants of the DE approach in Section 4. In Section 5, we will present the results obtained by applying the algorithms developed. In Section 6, we will discuss the results and provide suggestions for selecting an effective solution approach to the trust-based ridesharing problem. We conclude this paper in Section 7.

2. Literature Review

Although ridesharing is helpful to reduce the number of cars, energy consumption and cost, there are still concerns from drivers and riders in ridesharing. These concerns are the primary obstacles for ridesharing. According to [21,22], cost and time are two determinant factors for the adoption of ridesharing. The issue to maximize monetary incentive has been studied in [19]. Proper allocation of cost savings is an important issue in ridesharing systems [20]. In addition to the two determinant factors for the adoption of ridesharing, according to the review about rideshare crime rates and safety tips in [23], over 3000 sexual assaults were reported in one year for one major ridesharing company, not including unreported cases. The study in [24] indicates that women feel less safe and comfortable

when they share a ride in the night or with male strangers. In particular, young women and unemployed women have less trust in ridesharing. Therefore, how to ensure safety and trust in ridesharing systems is an important research issue to promote ridesharing.

One way to ensure safety/trust is to take advantage of the social relationship of ridesharing participants in relevant social networks. Social media provides a platform to extract the information of social networks. A recent survey on the challenges and opportunities of using social media to support ridesharing services is available in [6]. The types of ridesharing based on social networks are referred to as social ridesharing, sharing rides with friends or social-aware ridesharing in the literature. There are several studies relevant to social ridesharing. For example, in [7], the authors proposed a coalition formation algorithm for sharing rides with friends. A cooperative game-theoretic approach to the social ridesharing problem was proposed in [8]. The study of [8] considers a social network described by a graph to describe the social relationship between ridesharing participants. The social network restricts the formation of groups for creating a feasible coalition. A set of riders in the social network are a feasible coalition, if there exists a connected subgraph on the social network and there is at least one rider whose car has enough seats for all the members. However, the social network used in [8] does not consider the level of social relationship between ridesharing participants. In Ref. [9], the authors considered a social-aware ridesharing group query problem. The study of [9] indicates that the social-aware ridesharing group query problem is NP-hard. In Ref. [10], a relevant study of [9], an efficient method to match offers and requests in social-aware ridesharing was developed. A carpooling model based on both social and route networks was proposed in [11]. In Ref. [11], the degrees of separation concept and user preference are used to specify trust between ridesharing participants in the social networks. In Ref. [11], two problems are formulated: one problem aims to optimize trust in a ridesharing team based on social networks and the other problem is to minimize the cost for a ridesharing team with an integer programming model based on route networks. A decision model is obtained by combining the trust optimization problem and cost optimization problem. The concept of cohesive ridesharing in geo-social networks was studied in [12]. In Ref. [25], the authors take into account both social relations and revenue in the social-aware ridesharing group query problem.

The social network considered in the papers above is based on the social distance in social network model (an undirected graph). However, in the real world, the level that one participant trusts another may not be totally dependent on the social distance between them. For example, consider two people, A and B. Suppose A is a friend of B. But A also knows B often cannot be on time. The level that A trusts B is low. Suppose A is also a friend of C. Suppose the social distance between A and C is the same as that between A and B. A also knows C is always on time. The level that A trusts C is higher than the level that A trusts B. The above example indicates that although the level of trust between two participants may be linked to social distance, it may be negatively related to social distance between the two participants. A proper model to capture the level that one participant trusts another will be introduced in this paper.

Based on the discussion above, the goals of this study are to propose a decision model for ridesharing, based on a proper trust model, and develop a relevant solution methodology for solving the decision problem. For the decision problem to be addressed in this study, temporal constraints, spatial constraints and constraints due to trust requirements are considered. The decision model proposed in this paper aims to match multiple requests, which is different from the ones in [9,25]. The trust model used in this study can flexibly describe the level that one participant trusts another. In the proposed trust model, the trust level between two ridesharing participants may or may not be negatively related to social distance between the two participants. Therefore, the trust model adopted in this paper is different from the ones used in [9,11,25]. The decision model proposed in this study can be applied regardless of whether the level of trust between two participants is negatively related to their social distance.

The trust-based ridesharing problem is formulated as a non-linear integer programming problem, which consists of non-linear constraints and discrete decision variables. These problems are typically non-convex and the computational complexity grows with the scale of the problems. Therefore, an exact optimization method cannot be applied. Instead, approximate methods, such as evolutionary computation approaches, are adopted to find solutions. Evolutionary computation is inspired by biological evolution for global optimization. In evolutionary computation, a population of solutions is generated initially. The population of solutions is iteratively updated based on mutation and natural selection. Selection is based on the fitness function. Based on the iteratively updated solutions of population and natural selection, the quality of individuals in the population will be improved gradually. In the literature, a lot of evolutionary algorithms have been proposed to solve optimization problems. These include Differential Evolution [17], Particle Swarm Optimization [16] and Firefly algorithms [26], and their variants [27–29]. In this study, we will adopt several variants of Differential Evolution to develop algorithms to solve the trust-based ridesharing decision problem.

Many variants of DE algorithms have been proposed to improve the original DE method, including Differential Evolution with Neighborhood Search (NSDE) [30] and Self-adaptive Differential Evolution (SaDE) [31] and Self-adaptive NSDE (SaNSDE) [32], where SaNSDE is able to adapt the parameters of DE and combines the neighborhood search capability of NSDE. As the original DE and variants of DE algorithms are targeted at problems with continuous search space, these methods need to be modified for the discrete optimization problem formulated in this paper. In this paper, we will combine a self-adaptive neighborhood search with Differential Evolution to develop an algorithm variant of a discrete DE algorithm. In particular, we will study the influence of the learning period parameter on the efficiency of the proposed discrete SaNSDE algorithm. We will assess the effectiveness of the proposed algorithm by comparing with two Particle Swarm Optimization-based algorithms (standard PSO [13] and ALPSO [33]), a Firefly algorithm [26] and several DE algorithms, by applying these algorithms to the same set of test cases. We analyze the results of the experiments to compare the effectiveness of the algorithms mentioned above.

3. The Recommendation Problem of Trust-Based Shared Mobility Systems

In this section, we consider a shared mobility system in which trust requirements are considered. In shared mobility systems, drivers share rides with passengers to meet the transportation requirements. In this paper, a shared mobility system that considers the trust requirements of passengers and drivers is called a trust-based shared mobility system. A driver and a passenger can share a ride only if their trust requirements can be satisfied.

Although social network provides the social distance information that may be linked to safety or trust between people, in the real world, the level that one participant trusts another may not be totally dependent on the social distance between them. For example, consider two people, A and B, who are friends. Suppose A is always on time whereas B is rarely on time. The level that A trusts B is low but the level that B trusts A is high. The above example indicates that shorter social distance in the social network does not always imply a higher level of trust. That is, social distance may not always be negatively related to the level of trust. A proper trust model to capture the level that one participant trusts another is used in this study. The trust model used in this study can flexibly describe the level that one participant trusts another. In the proposed trust model, the trust level between two ridesharing participants may or may not be negatively related to social distance between the two participants. The solution methodology proposed in this study can still be applied, regardless of whether the trust model is the same as the social distance model. To describe the decision model in this paper, we summarize the notations used in this paper in Table 1.

Table 1. Notations of symbols, variables and parameters.

Variable	Meaning
P	total number of potential passengers
D	total number of potential drivers
K	total number of locations of passengers, K
p	a passenger, where $p \in \{1, 2, 3, \ldots P\}$
d	a driver, where $d \in \{1, 2, 3, \ldots, D\}$
k	a location, $k \in \{1, 2, \ldots, K\}$
J_d	total bids of driver $d \in \{1, 2, \ldots, D\}$
j	The $j-th$ bid of driver d with $j \in \{1, 2, \ldots, J_d\}$
Γ_d	the minimal trust level requested by driver$d$
Λ_p	the minimal trust level requested by passenger p
V	the set of nodes associated with drivers and passengers in the social network model
v_i	a node in V
E	the set of edges in the social network model
e_{ij}	a directed edge in E connecting v_i to v_j, where $v_i, v_j \in V$
Θ	a $\lvert V\rvert$ by $\lvert V\rvert$ matrix with element Θ_{ij} for all $v_i, v_j \in V$, where Θ_{ij} is the trust level that v_i trusts v_j.
$S(V, E, \Theta)$	a graph model of the social network with a set of nodes, V, a set of edges and weight Θ.
R_p	a passenger's request; $R_p = (Lo_p, Le_p, \omega_p^e, \omega_p^l, n_p, \Lambda_p)$, where R_p is defined by the origin Lo_p, destination, Le_p, earliest departure time, ω_p^e, latest arrival time, ω_p^l, number of passengers, n_p, and the minimal trust level requested, Λ_p.
R_d	a driver's request; $R_d = (Lo_d, Le_d, \omega_d^e, \omega_d^l, a_d, \overline{\tau}_d, \Gamma_d)$, where R_d is defined by the origin, Lo_d, destination, Le_d, earliest departure time, ω_d^e, latest arrival time, ω_d^l, quantity of seats available, a_d, maximum detour ratio, $\overline{\tau}_d$, and the minimal trust level Γ_d. That is, $R_d = (Lo_d, Le_d, \omega_d^e, \omega_d^l, a_d, \overline{\tau}_d, \Gamma_d)$.
D_BID_{dj}	$D\_BID_{dj} = (q_{dj1}^1, q_{dj2}^1, \ldots, q_{djk}^1, \ldots, q_{djK}^1, q_{dj1}^2, q_{dj2}^2, \ldots, q_{djk}^2, \ldots, q_{djK}^2, \pi_{dj}, o_{dj}, c_{dj}, a_d, \Gamma_d)$, the $j-th$ bid of driver d, where K is the number of locations, q_{djk}^1 is the number of seats available to pick up passengers at location k, q_{djk}^2 is the number of seats released after dropping passengers at location $P+k$, o_{dj} is the original cost of driver d without ridesharing, c_{dj} is the cost for driver d to transport passengers in the bid, a_d is the total number of seats and Γ_d is the minimal trust level requested.
P_BID_p	$P\_BID_p = (s_{p1}^1, s_{p2}^1, s_{p3}^1, \ldots, s_{pK}^1, s_{p1}^2, s_{p2}^2, s_{p3}^2, \ldots, s_{pK}^2, f_p, \Lambda_p)$: the bid of passenger p, where K is the number of locations, s_{pk}^1 is the number of seats requested to pick up passengers at location k, s_{pk}^2 is the number of seats released after dropping passengers at location $P+k$, f_p is the bid price and Λ_p is the minimal trust level requested.
x_{dj}	a binary decision variable; x_{dj} equals 1 if the $j-th$ bid of driver d is a winning bid and x_{dj} equals 0 otherwise
y_p	a binary decision variable; y_p equals 1 if the bid of passenger p is a winning bid and y_p equals 0 otherwise
$F(x, y)$	overall cost savings, $F(x,y) = \left(\sum_{p=1}^{P} y_p\left(f_p\right)\right) + \left(\sum_{d=1}^{D}\sum_{j=1}^{J_d} x_{dj} o_{dj}\right) - \left(\sum_{d=1}^{D}\sum_{j=1}^{J_d} x_{dj} c_{dj}\right)$

Trust between drivers and passengers is described based on their social network. The trust requirements are directly related to the connection of drivers and passengers in the social network. A graph-based model is adopted in this study to represent the social network. To represent drivers and passengers in the social network model, we use V to denote the set of nodes associated with drivers and passengers. The trust level between nodes is represented by weight, associated with the set of edges, E, in the social network model. Let v_i and v_j be two nodes in V. We use e_{ij} to denote a directed edge e_{ij} connecting v_i to v_j and use weight Θ_{ij} to denote the trust relation between v_i and v_j, where the weight Θ_{ij} specifies the degree (trust level) that v_i trusts v_j. As the goal of this study is to develop a decision model and relevant solution methodology, it is assumed that Θ_{ij} is available.

Let Θ denote the $|V|$ by $|V|$ matrix with element Θ_{ij} for all $v_i, v_j \in V$. The greater the value of Θ_{ij}, the more v_i trusts v_j. If Θ_{ij} equals zero, v_i does not trust v_j. More specifically, there is a directed edge e_{ij} connecting v_i to v_j with weight Θ_{ij} for each v_i and v_j in V. We use a graph $S(V, E, \Theta)$ with weight Θ to compactly represent the trust model.

In a trust-based shared mobility system, a driver may request his/her minimal trust level requirements to share a ride with a passenger. If the minimal trust level requirements

requested by the driver cannot be satisfied, the driver will not share a ride with the passenger. Similarly, a passenger may request his/her minimal trust level requirements to share a ride with a driver. If the minimal trust level requirements requested by the passenger cannot be satisfied, the passenger will not share a ride with the driver. Furthermore, a passenger, say A, may request his/her minimal trust level requirements to share a ride with another passenger, say B. If the minimal trust level requirements requested by passenger A cannot be satisfied, passenger A will not share a ride with passenger B.

To represent the above trust level requirements requested by drivers and passengers, we use Γ_d to denote the minimal trust level requested by driver d and Λ_p to denote the minimal trust level requested by passenger p. A driver d will share a ride with passenger p only if $\Theta_{dp} \geq \Gamma_d$. A passenger p can be a rider with driver d only if $\Theta_{pd} \geq \Lambda_p$. A passenger p can be a rider with another passenger p' only if $\Theta_{pp'} \geq \Lambda_p$.

Potential drivers and passengers express their transportation requirements and trust requirements by submitting requests to the shared mobility system. The shared mobility system needs to determine the drivers and passengers for ridesharing.

To formulate the decision problem, information from the requests submitted by drivers and passengers are briefly described first. A request of passenger p is denoted by R_p, where R_p is defined by the origin Lo_p, destination, Le_p, earliest departure time, ω_p^e, latest arrival time, ω_p^l, number of passengers, n_p, and the minimal trust level requested, Λ_p. That is, $R_p = (Lo_p, Le_p, \omega_p^e, \omega_p^l, n_p, \Lambda_p)$.

A driver's request is denoted by R_d, where R_d is defined by the origin, Lo_d, destination, Le_d, earliest departure time, ω_d^e, latest arrival time, ω_d^l, quantity of seats available, a_d, maximum detour ratio, $\overline{\tau}_d$, and the minimal trust level Γ_d. That is, $R_d = (Lo_d, Le_d, \omega_d^e, \omega_d^l, a_d, \overline{\tau}_d, \Gamma_d)$.

Let P be the total number of potential passengers that submit requests to the shared mobility system and let $\{1, 2, \ldots, P\}$ be the set of all potential passengers. Without loss of generality, it is assumed that there is only one request, R_p, submitted by each passenger $p \in \{1, 2, \ldots, P\}$. For each passenger $p \in \{1, 2, \ldots, P\}$, a procedure will be invoked by the shared mobility system to generate bid $P\_BID_p = (s_{p1}^1, s_{p2}^1, s_{p3}^1, \ldots, s_{pK}^1, s_{p1}^2, s_{p2}^2, s_{p3}^2, \ldots, s_{pK}^2, f_p, \Lambda_p)$ based on R_p, where K is the number of locations, s_{pk}^1 is the number of seats requested to pick up passengers at location k, s_{pk}^2 is the number of seats released after dropping passengers at location $P + k$, f_p is the bid price and Λ_p is the minimal trust level requested.

Let D denote the total number of potential drivers that submit requests to the shared mobility system and let $\{1, 2, \ldots, D\}$ be the set of all potential drivers. Without loss of generality, it is assumed that there is only one request, R_d, will be submitted by each driver $d \in \{1, 2, \ldots, D\}$. For each driver $d \in \{1, 2, \ldots, D\}$, a procedure will be invoked by the shared mobility system to generate J_d bids $D\_BID_{dj} = (q_{dj1}^1, q_{dj2}^1, \ldots, q_{djk}^1, \ldots, q_{djK}^1, q_{dj1}^2, q_{dj2}^2, \ldots, q_{djk}^2, \ldots, q_{djK}^2, \pi_{dj}, o_{dj}, c_{dj}, a_d, \Gamma_d)$ based on R_d, where J_d is the total number of bids of a driver d, j is the $j - th$ bid of driver d with $j \in \{1, 2, \ldots, J_d\}$, K is the number of locations, q_{djk}^1 is the number of seats available to pick up passengers at location k, q_{djk}^2 is the number of seats released after dropping passengers at location $P + k$, o_{dj} is the original cost of driver d without ridesharing, c_{dj} is the cost for driver d to transport passengers in the bid, a_d is the total number of seats and Γ_d is the minimal trust level requested.

Based on the bids, P_BID_p, $p \in \{1, 2, \ldots, P\}$ and D_BID_{dj}, $d \in \{1, 2, \ldots, D\}$, $j \in \{1, 2, \ldots, J_d\}$, we define decision variables, $x_{dj} \forall d \in \{1, \ldots, D\}$ $\forall j \in \{1, \ldots, J_d\}$ for the bids submitted by drivers. The j-th bid placed by driver d is a winning bid if $x_{dj} = 1$. Otherwise, $x_{dj} = 0$. We define decision variables as, $y_p \forall p \in \{1, 2, 3, \ldots P\}$. The bid submitted by passenger p is a winning bid if $y_p = 1$ and is not a winning bid if $y_p = 0$. We define the objective function defined in (1) to maximize the cost savings, where $F(x, y) = \sum_{p=1}^{P} y_p f_p + \sum_{d=1}^{D} \sum_{j=1}^{J_d} x_{dj}(o_{dj} - c_{dj})$.

We formulate the problem for trust-based shared mobility systems considering the constraints that the number of seats supplied and the number of seats requested at each pick-up location must be the same (defined in (2)), the number of seats released must be equal to the number of passengers dropped off at each drop-off location (defined in (3)), the cost savings must be nonnegative (defined in (4)), each driver can have at most one bid accepted (defined in (5)), the minimal trust level requested by each driver must be satisfied (defined in (6)) and the minimal trust level requested by each passenger must be satisfied (defined in (7)). The decision variables must be binary (defined in (8)). The problem formulation is as follows:

$$\max_{x,y} F(x,y) \quad s.t. \tag{1}$$

$$\sum_{d=1}^{D}\sum_{j=1}^{J_d} x_{dj} q^1_{djk} = y_p s^1_{pk} \; \forall p \in \{1,2,\ \ldots, P\} \;\; \forall \mathrm{k} \in \{1,2,\ \ldots, P\} \tag{2}$$

$$\sum_{d=1}^{D}\sum_{j=1}^{J_d} x_{dj} q^2_{djk} = y_p s^2_{pk} \; \forall p \in \{1,2,\ \ldots, P\} \;\; \forall \mathrm{k} \in \{1,2,\ \ldots, P\} \tag{3}$$

$$\sum_{p=1}^{P} y_p f_p + \sum_{d=1}^{D}\sum_{j=1}^{J_d} x_{dj} o_{dj} \geq \sum_{d=1}^{D}\sum_{j=1}^{J_d} x_{dj} c_{dj} \tag{4}$$

$$\sum_{j=1}^{J_d} x_{dj} \leq 1 \; \forall d \in \{1,\ldots,D\} \tag{5}$$

$$\sum_{k=1}^{K} q_{djk} s_{pk} x_{dj} y_p (\Theta_{dp} x_{dj} y_p - \Gamma_d) \geq 0 \quad \forall d \in \{1,\ldots,D\} \forall j \in \{1,\ldots,J_d\} \forall p \in \{1,2,3,\ldots P\} \tag{6}$$

$$\sum_{k=1}^{K} q_{djk} s_{pk} x_{dj} y_p (\Theta_{dp} x_{dj} y_p - \Lambda_p) \geq 0 \quad \forall d \in \{1,\ldots,D\} \; \forall j \in \{1,\ldots,J_d\} \; \forall p \in \{1,2,3,\ldots P\} \tag{7}$$

$$x_{dj} \in \{0,1\} \forall d \in \{1,\ldots,D\} \; \forall j \in \{1,\ldots,J_d\}, y_p \in \{0,1\} \; \forall p \in \{1,2,\ \ldots, P\} \tag{8}$$

4. Development of a Self-Adaptive Neighborhood Search Differential Evolution Algorithm

In this section, we will present the proposed algorithm. Table 2 lists the symbols and notations used in this section. In the development of an evolutionary computation algorithm, a fitness function must be properly designed to guide the evolution process to find a solution. A proper fitness function needs to take into consideration the objective function of the problem and the constraints that need to be satisfied. A properly designed fitness function has to guide the evolution process based on constraint violations. We will apply a variant of penalty methods to design a fitness function.

Table 2. Notations of symbols, variables and parameters used in the proposed algorithm.

Variable	Meaning
NP	population size
G	number of generations
g	the index of a generation
LP	learning period
F_i	scale factor of individual i
cr_i	crossover rate of individual i
f_p	a parameter to influence the scale factor F_i
s	a mutation strategy : s= 1 denotes mutation strategy in (9) and s = 2 denotes mutation strategy in (13)
n_1	the number of individuals (offspring) generated by mutation strategy in (9) that successfully replace the original individual and enter the next generation
m_1	the number of individuals(offspring) generated by mutation strategy in (9) that fail to replace the original individual and are discarded
n_2	the number of individuals(offspring) generated by mutation strategy in (13) that successfully replace the original individual and enter the next generation
m_2	the number of individuals(offspring) generated by mutation strategy in (13) that fail to replace the original individual and are discarded
f_p	a parameter to influence the scale factor F_i, $f_p = \frac{n_1(n_2+m_2)}{n_2(n_1+m_1)+n_1(n_2+m_2)}$
CR_{rec}	an array that records the crossover rate value cr_i associated with individual i that enter the next generation
CR_m	a parameter to generate the cross over rate cr_i of individual i defined by $CR_m = \frac{\sum_{k=1}^{\lvert CR_{rec}\rvert} CR_{rec}(k)}{\lvert CR_{rec}\rvert}$
r	A random variable with uniform distribution $U(0,1)$
r_1	A random variable with Gaussian distribution $N(\mu, \sigma_1^2)$ with mean μ and standard deviation σ_1
r_2	A random variable r with uniform distribution $U(0,1)$

The problem formulated in the previous section is an integer programming optimization problem, in which the decision variables are binary. A characteristic of this problem is the exponential growth of solution space with problem size and constraints. The exponentially growing solution space leads to high complexity from a computational point of view. In the literature, many variants of classical penalty methods have been proposed to deal with constraints in constrained optimization problems by applying penalty methods. Although these methods may work under the proper setting of penalty coefficients, they suffer from parameter tuning problems that are problem/data dependent. It is hard to find a one-size-fits-all approach to finding the right parameters for classical penalty methods. The variant of a penalty method, proposed in [34] and adopted in this paper, is different from the classical penalty methods in that it does not require any penalty parameter. It applies pair-wise comparison between two solutions in the optimization processes to select the better solution in the following way. If both solutions to be compared are feasible, the one with better objective function value will be selected. If one feasible solution and another infeasible solution is to be compared, the feasible one will be selected. If both solutions to be compared are infeasible, the one with a smaller constraint violation will be selected. This approach effectively reduces the constraint violation in the optimization processes and makes the solution found move towards the feasible region.

We adopt the method proposed in [34] to differentiate feasible solutions and infeasible ones. Just like classical penalty methods, this approach also captures the effects of constraint violation by adding several penalties. To describe the above method, we define the following notations. Let $S_f = \{(x,y)|(x,y)\}$, a solution in the current population, (x,y) satisfies constraints (1)–(8). S_f is the set of all feasible solutions in the current population. Let $S_{f\min}$ be the object function value of the worst feasible solution in the current population. More formally, $S_{f\min} = \min_{(x,y)\in S_f} F(x,y)$.

The fitness function used in this paper is defined by $F_1(x,y)$ as follows: $F_1(x,y) = \begin{cases} F(x,y)\ if\ (x,y)\ satisfies\ constraints\ (1) \sim (8) \\ U(x,y)\ otherwise \end{cases}$, where

$$
\begin{aligned}
&U(x,y) = S_{f\min} + U_1(x,y) + U_2(x,y) + U_3(x,y) + U_4(x,y) + U_5(x,y) \\
&U_1(x,y) = \left| \sum_{p=1}^{P} \sum_{k=1}^{K} \left(\sum_{d=1}^{D} \sum_{j=1}^{J_d} x_{dj} q_{djk} - y_p s_{pk} \right) \right| \\
&U_2(x,y) = \left| \sum_{d=1}^{D} \left(1 - \sum_{j=1}^{J_d} x_{dj}\right) \right| \\
&U_3(x,y) = \min\left(\sum_{p=1}^{P} y_p f_p + \sum_{d=1}^{D} \sum_{j=1}^{J_d} x_{dj} o_{dj} - \sum_{d=1}^{D} \sum_{j=1}^{J_d} x_{dj} c_{dj}, 0.0 \right) \\
&U_4(x,y) \\
&= \sum_{d=1}^{D} \sum_{j=1}^{J_d} \sum_{p=1}^{P} \min\left(\sum_{k=1}^{K} q_{djk} s_{pk} \right) x_{dj} y_p (\Theta_{dp} x_{dj} y_p - \Gamma_d), 0.0) \\
&U_5(x,y) \\
&= \sum_{d=1}^{D} \sum_{j=1}^{J_d} \sum_{p=1}^{P} \min\left(\sum_{k=1}^{K} q_{djk} s_{pk} x_{dj} y_p (\Theta_{dp} x_{dj} y_p - \Lambda_p), 0.0 \right)
\end{aligned}
$$

As the proposed SaNSDE algorithm is based on an extension of the standard DE algorithm, a brief introduction to the standard DE algorithm will be presented first. The details of the SaNSDE algorithm will be described next.

The standard DE algorithm follows a four-step evolution process to iteratively improve the quality of solutions. The first step of the standard DE algorithm is initialization. Suppose the problem dimension is L. In the initialization step, the standard DE algorithm creates a population of NP trial individuals $z_{gi} = (z_{gil})$, where g is the generation index, i is the index of an individual, $i \in \{1, 2, \ldots, NP\}$, and $l \in \{1, 2, \ldots, L\}$ is the dimension index. In the second step, the DE algorithm generates a mutant vector $v_{gi} = (v_{gil})$ for individual i in the g-th generation by applying a mutation strategy. There are six well-known mutation strategies for the standard DE algorithm in the literature. These six mutation strategies are defined based on the best individual $Z_{gbl} = (z_{gbl})$ and the randomly selected individuals, z_{gr_1l}, z_{gr_2l}, z_{gr_3l}, z_{gr_4l} and z_{gr_5l}, in the population, where r_1, r_2, r_3, r_4 and r_5 are random integers between 1 and NP. The six mutation strategies are defined as follows:

$$v_{(g+1)il} = z_{gr_1l} + F_i(z_{gr_2l} - z_{gr_3l}) \tag{9}$$

$$v_{(g+1)il} = z_{gbl} + F_i(z_{gr_2l} - z_{gr_3l}) \tag{10}$$

$$v_{(g+1)il} = z_{gr_1l} + F_i(z_{gr_2l} - z_{gr_3l}) + F_i(z_{gr_4l} - z_{gr_5l}) \tag{11}$$

$$v_{(g+1)il} = z_{gbl} + F_i(z_{gr_1l} - z_{gr_2l}) + F_i(z_{gr_3l} - z_{gr_4l}) \tag{12}$$

$$v_{(g+1)il} = z_{gil} + F_i(z_{gbl} - z_{gil}) + F_i(z_{gr_1l} - z_{gr_2l}) \tag{13}$$

$$v_{(g+1)il} = z_{gil} + F_i(z_{gbl} - z_{gil}) + F_i(z_{gr_1l} - z_{gr_2l}) + F_i(z_{gr_3l} - z_{gr_4l}) \tag{14}$$

The third step is a crossover operation that follows after a mutant vector v_{gi} is generated. The crossover operation will create a trial vector u_{gi} by selecting the element between the mutant vector v_{gi} and the original individual z_{gi} with some probability. The last step is selection. In the selection step, z_{gi} will be replaced by u_{gi} if u_{gi} is better than z_{gi}.

In the proposed Self-adaptive Neighborhood Search Differential Evolution (SaNSDE) Algorithm, the neighborhood search concept is combined with the self-adaptation of mutation strategies and scale factor to search solutions. The pseudocode of the SaNSDE algorithm is shown in Table 3. In SaNSDE, there are two algorithmic parameters, f_p and CR_m. These two parameters will be self-adapted to influence the scale factor and the crossover rate of DE. The scale factor F_i is determined randomly based on the parameter f_p. The crossover rate cr_i of individual i is generated randomly according to the parameter CR_m.

Table 3. The pseudo code of SaNSDE Algorithm.

Discrete Self-Adaptive Neighborhood Search Differential Evolution with (SaNSDE) Algorithm

Step 0: Initialize a population with NP individuals randomly
$CR_m = 0.5$
$f_p = 0.5$
For $g = 1$ *to* G
For $i = 1$ *to* NP
Generate a random variable r with uniform distribution $U(0,1)$
If $r < f_p$
Generate a random variable r_1 with Gaussian distribution $N(\mu, \sigma_1^2)$
$F_i = r_1$
Else
Generate a random variable r_2 with uniform distribution $U(0,1)$
$F_i = r_2$
End If
Generate a random variable cr_i with Gaussian distribution $N(CR_m, \sigma_2^2)$
Step 1: Create a mutant vector v_{gi}
Generate $rand_i = U(0,1)$
If $rand_i < f_p$
$s = 1$
Calculate v_{gi} according to (9)
Else
$s = 2$
Calculate v_{gi} according to (13)
End If
Step 2: Create a trial vector u_{gi}
For $l \in 1, 2, \ldots, L$
$u_{gil} = \begin{cases} v_{gil} \; if \; Rand(0,1) < cr_i \\ z_{gil} \; otherwise \end{cases}$
$\overline{u}_{gil} \leftarrow BinaryTrasform(u_{gil})$
End For
Step 3: Select the trial vector if it outperforms individual i
If $F_1(\overline{u}_{gi}) \geq F_1(z_{gi})$
$z_{(g+1)i} = u_{gi}$
Record cr_i in CR_{rec}
$n_s = n_s + 1$
Else
$m_s = m_s + 1$
End If
End For
If $g > LP$
$f_p = \frac{n_1(n_2+m_2)}{n_2(n_1+m_1)+n_1(n_2+m_2)}$
$CR_m = \frac{\sum_{k=1}^{|CR_{rec}|} CR_{rec}(k)}{|CR_{rec}|}$
End If
End For

The SaNSDE algorithm keeps track of four variables, n_1, n_2, m_1 and m_2, to adapt the parameter f_p as follows:

$f_p = \frac{n_1(n_2+m_2)}{n_2(n_1+m_1)+n_1(n_2+m_2)}$, where n_1, n_2, m_1 and m_2 are defined as follows:

n_1: the number of individuals (offspring) generated by mutation strategy in (9) that successfully replace the original individual and enter the next generation,

m_1: the number of individuals (offspring) generated by mutation strategy in (9) that fail to replace the original individual and are discarded,

n_2: the number of individuals (offspring) generated by mutation strategy in (13) that successfully replace the original individual and enter the next generation,

m_2: the number of individuals (offspring) generated by mutation strategy in (13) that fail to replace the original individual and are discarded.

In the SaNSDE algorithm, it records the crossover rate value cr_i associated with individual i that enter the next generation in an array CR_{rec} to adapt the parameter CR_m. Adaptation of the parameter CR_m is done by the following formula:

$$CR_m = \frac{\sum_{k=1}^{|CR_{rec}|} CR_{rec}(k)}{|CR_{rec}|}.$$

As the solution space of the decision problem is discrete, a procedure, $BinaryTrasform$, is used to transform the continuous decision variables to zero or one. The fitness function is computed based on the transformed binary values of decision variables.

Procedure $BinaryTrasform$
Input: a
Output: c
Begin

$$b = \begin{cases} V_{\max} & if\ a > V_{\max} \\ a & if\ -V_{\max} \le a \le V_{\max} \\ -V_{\max} & if\ a < -V_{\max} \end{cases}$$

$s(b) = \frac{1}{1+\exp^{-b}}$
Generate a random variable $rsid$ with uniform distribution $U(0,1)$

$$c = \begin{cases} 1\ rsid < s(b) \\ 0\ otherwise \end{cases}$$

Return c

Based on the notations defined above, the proposed algorithm is shown in Table 3.

The underlying principle of the SaNSDE algorithm is very simple and intuitive. The learning period (LP) parameter is used to specify the number of generations to learn the effectiveness of different strategies. During the learning period, the SaNSDE algorithm attempts to apply different strategies to solve the problem and collects the data based on the solution found to calculate the probability to select different strategies. If a strategy s applied can lead to improvement in solution quality, the strategy success counter n_s will be increased by one. Otherwise, the failure counter m_s will be increased by one. After the learning period, the success counter n_s and the failure counter m_s will be used to calculate f_p, which will influence the probability to select different strategies in Step 1. The strategy with the higher success count will have higher probability to be selected after the learning period.

5. Results

The purpose of this section is twofold: (1) to verify the functionality to meet trust requirements of drivers and passengers, (2) to study the characteristics of the proposed algorithm and (3) to assess the effectiveness of the proposed algorithm. For the first part, we apply the proposed algorithm to an example of the trust-based ridesharing problem to assess its functionality to satisfy the trust requirements of drivers and passengers, based on the results of the algorithm. For the second part, we study the influence of the learning period on the performance and efficiency of the proposed algorithm. For the third part, we study the effectiveness of the proposed algorithm by performing a number of experiments based on several test cases and analyze the numerical results of the experiments.

5.1. Two Scenarios

We consider two scenarios with different trust requirements to illustrate the influence of trust requirements on ridesharing. In Scenario 1 and Scenario 2, below, we consider the same set of three drivers and ten passengers in a trust-based ridesharing system. However, some of the trust requirements are not the same.

The data for these two scenarios can be downloaded from the following link: https://drive.google.com/drive/folders/1W4OdMQ6z8O0fX38TpSVFcVbsbzJkjHy3?usp=sharing (accessed on 20 January 2022).

5.1.1. Scenario 1

The origins and destinations of the drivers and passengers are listed in Table 4.

Table 4. Origins and Destinations of Participants.

Participant	Origin	Destination
Driver 1	24.23785, 120.66993	24.11308, 120.65914
Driver 2	24.1692536, 120.6848233	24.20195, 120.56815
Driver 3	24.13425, 120.5539	24.14289, 120.70549
Passenger1	24.21872, 120.6469	24.12877, 120.66223
Passenger2	24.1790507, 120.6657476	24.17369, 120.61354
Passenger3	24.0611, 120.64342	24.1465287, 120.6532456
Passenger4	24.07962, 120.69454	24.12438, 120.66244
Passenger5	24.19422, 120.69538	24.15288, 120.69704
Passenger6	24.16048, 120.69173	24.16359, 120.65138
Passenger7	24.0611, 120.64342	24.11009, 120.64146
Passenger8	24.19422, 120.69538	24.13046, 120.7047
Passenger9	24.13623, 120.60693	24.13527, 120.6571
Passenger10	24.16429, 120.68522	24.15345, 120.65495

The minimal trust level Γ_d requested by each driver d is listed in Table 5. The minimal trust level Λ_p requested by each passenger p is listed in Table 6. The trust level between driver and passenger is represented by the matrix Θ, shown in Table 7.

Table 5. Γ_d for Drivers.

Driver ID (*d*)	**1**	**2**	**3**
Γ_d	1	1	3

Table 6. Λ_p for Passengers.

Passenger ID (*d*)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Λ_p	1	1	1	1	1	1	1	1	1	1

Table 7. Matrix Θ.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	1	3	2	3	1	2	3	2
2	2	1	1	1	2	2	3	3	1	2
3	1	2	3	2	2	3	3	1	1	2

Tables 8 and 9 show the bids of all drivers and passengers, respectively, generated by the bid generation procedure in Appendix II [31].

Table 8. Bid submitted by Driver 1.

Driver ID (d)	q_{d11}	q_{d12}	q_{d13}	q_{d14}	q_{d15}	q_{d16}	q_{d17}	q_{d18}	q_{d19}	q_{d110}	o_{d1}	c_{d1}
1	0	0	0	0	1	0	0	0	0	0	50.4025	51.4975
2	0	0	0	0	0	0	0	0	0	1	36.745	41.1575
3	0	0	0	0	0	0	0	0	1	0	57.485	57.485

Table 9. Bids submitted by Passengers.

Passenger ID (p)	s_{p1}	s_{p2}	s_{p3}	s_{p4}	s_{p4}	s_{p4}	s_{p4}	s_{p4}	s_{p4}	s_{p4}	f_p
1	1	0	0	0	0	0	0	0	0	0	37.0475
2	0	1	0	0	0	0	0	0	0	0	18.3475
3	0	0	1	0	0	0	0	0	0	0	28.12
4	0	0	0	1	0	0	0	0	0	0	19.07
5	0	0	0	0	1	0	0	0	0	0	14.1675
6	0	0	0	0	0	1	0	0	0	0	12.25
7	0	0	0	0	0	0	1	0	0	0	17.1175
8	0	0	0	0	0	0	0	1	0	0	33.11
9	0	0	0	0	0	0	0	0	1	0	14.6925
10	0	0	0	0	0	0	0	0	0	1	9.645

We apply the proposed SaNSDE algorithm to solve the decision problem using the following parameters:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
$F_i = 0.3r_1 + 0.5$, where r_1 is a Gaussian distribution $N(0, 1)$.
LP = 10, 50, 100, 200, 1000.
$V_{\max} = 4$.

Based on the above input data, the solution found by applying the SaNSDE algorithm is shown in Tables 10 and 11.

Table 10. Solution x for Drivers.

Driver ID (d)	x_{11}	x_{21}	x_{31}
1	1	1	0

Table 11. Solution y for Passengers.

Passenger ID (d)	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
1	0	0	0	0	1	0	0	0	0	1

The cost savings for this solution are 18.305.

The two shared routes generated by the solution found by applying the SaNSDE algorithm for Example 1 are shown in Figure 1. Please visit the following link to display the routes of Driver 1 and Driver 2 and the associated ridesharing passengers. https://www.google.com/maps/d/edit?mid=1_i--xWSlx6_a-ISgTLMzKP8_PPzjYACH&usp=sharing (accessed on 20 January 2022).

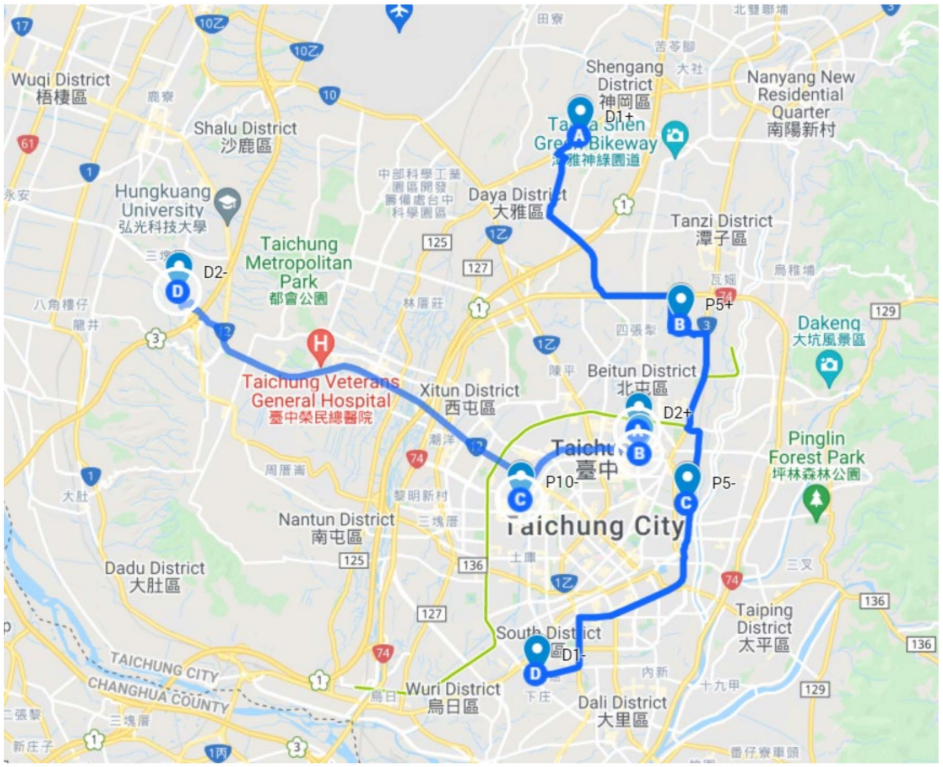

Figure 1. Two shared routes for Case 1 of the Example.

5.1.2. Scenario 2

In Scenario 2, the itineraries of drivers and passengers are the same as those of Scenario 1. However, the trust requirements between drivers and passengers in Scenario 2 are different from those of Scenario 1. We will study the influence of trust requirements between drivers and passengers on the number of matched shared rides.

As the itineraries of drivers and passengers in Scenario 2 are the same as those of Scenario 1, the origins and destinations of the drivers and passengers in Scenario 2 are the same as the information listed in Table 4.

The minimal trust level Γ_d requested by each driver d is listed in Table 12. The minimal trust level Λ_p requested by each passenger p is listed in Table 13. The trust level between driver and passenger is represented by the matrix Θ, shown in Table 14.

Table 12. Γ_d for Drivers.

Driver ID (*d*)	**1**	**2**	**3**
1	1	3	3

Table 13. Λ_p for Passengers.

Passenger ID (*d*)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Λ_p	1	1	1	1	1	1	1	1	1	1

Table 14. Matrix Θ.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	1	3	2	3	1	2	3	2
2	2	1	1	1	2	2	3	3	1	2
3	1	2	3	2	2	3	3	1	1	2

We apply the proposed SaNSDE algorithm to solve the decision problem using the same parameters used in Scenario 1.

Based on the above input data, the solution found by applying the SaNSDE algorithm is shown in Tables 15 and 16. The results of Scenario 2 indicate that Γ_d has influence on the number of matched shared rides. The results of Scenario 2 show that the number of ridesharing passengers has been reduced to one. This is due to the constraints of the trust requirement set by driver two.

Table 15. Solution x for Drivers.

Driver ID (d)	x_{11}	x_{21}	x_{31}
1	1	0	0

Table 16. Solution y for Passengers.

Passenger ID (d)	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
1	0	0	0	0	1	0	0	0	0	0

The cost savings for this solution are 13.07.

Two shared routes for Case 2 of Example 1 are shown in Figure 2.

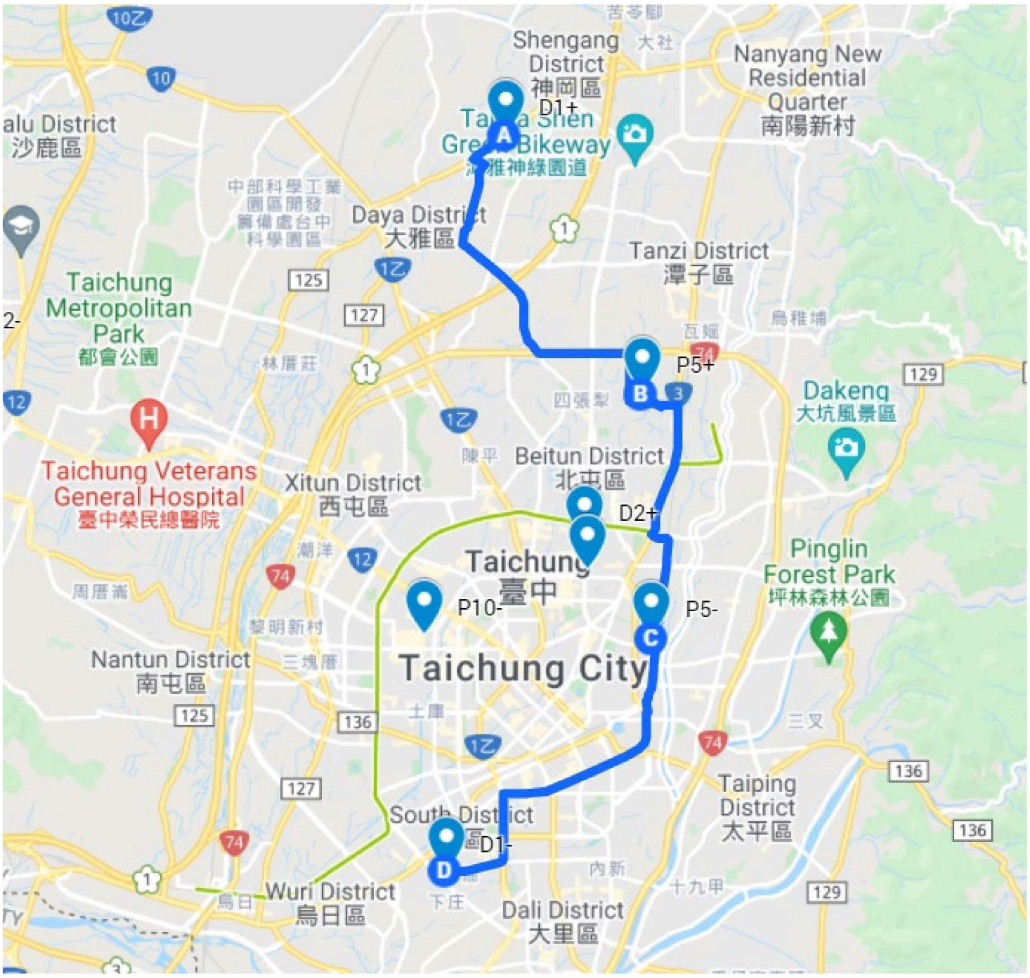

Figure 2. The shared route for Case 2 of the Example.

Please visit the following link to display the routes of Driver 1 and Driver 2 and the associated ridesharing passengers. https://www.google.com/maps/d/edit?mid=1l_XW_SzWAu3hkvkS5pLrJF3XPqrm6xNh&usp=sharing (accessed on 20 January 2022).

5.2. Influence of Learning Period on Performance and Efficiency

There is an algorithmic parameter, learning period (LP), in the proposed SaNSDE algorithm. To apply the proposed method to solve problems, we first study the influence of the algorithmic parameter LP on convergence speed.

The parameters used in the Self-adaptive Neighborhood Search Differential Evolution (SaNSDE) algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
$F_i = 0.3r_1 + 0.5$, where r_1 is a Gaussian distribution $N(0, 1)$.
LP = 10, 50, 100, 200, 1000.
$V_{\max}$ = 4.

Table 17 shows the results obtained by setting the algorithmic parameter LP to 10, 50, 100, 200 and 1000, respectively. For each test case, the average fitness values obtained by setting LP to 100, 200 and 1000 are very close. However, the average number of generations required to find the best solutions are different for different algorithmic parameters LP. The average number of generations for Case 1, Case 2, Case 3, Case 4 and Case 5 are shown in the bar chart of Figure 3. The average number of generations for Case 6, Case 7, Case 8, Case 9 and Case 10 are shown in the bar chart of Figure 4. The results indicate that the average number of generations required to find the best solutions tend to decrease by increasing LP for most test cases. This is due to the quality of the best strategy being improved with the growth of the algorithmic parameter LP. If the algorithmic parameter LP is too small, the best strategy cannot be learned within the learning period. In this case, the probability to select the best strategy to create a mutant vector will be lower than expected. Therefore, the number of generations required to find the best solutions will be greater. If the algorithmic parameter LP is big enough, the best strategy can be learned within the learning period. In this case, the probability to select the best strategy to create a mutant vector will be higher. Therefore, the number of generations required to find the best solutions will be smaller. Based on the above reasoning, the results in Figures 3 and 4 are consistent with our expected results for almost all test cases, with the exception of Case 9. This is due to the fact that the SaNSDE algorithm is a stochastic optimization method.

Table 17. Average fitness values and average number of generations for SaNSDE with algorithmic parameter LP to 10, 50, 100, 200 and 1000.

Test Case	Participant (D/P)	SaNSDE (LP = 10)	SaNSDE (LP = 50)	SaNSDE (LP = 100)	SaNSDE (LP= 200)	SaNSDE (LP = 1000)
1	3/10	18.305/100.3	18.305/32.9	18.305/9.9	18.305/6.4	18.305/2.8
2	5/11	23.518/236.9	23.518/39.3	23.518/21.8	23.518/14.9	23.518/7.6
3	5/12	24.79/288.7	24.79/39.5	24.79/17.6	24.79/15.3	24.79/7.1
4	6/12	36.58/752.8	36.58/127	36.58/53.7	36.58/27.1	36.58/12.9
5	7/13	29.2442/1426.1	30.063/50.9	30.063/92.3	30.063/31.6	30.063/14.7
6	8/14	62.18/116	62.18/38	62.18/22.9	62.18/18.1	62.18/7
7	9/15	74.0355/1746.9	79.64/2055.2	79.64/609.1	79.64/200.4	79.64/45.4
8	20/20	23.644/2582.2	23.644/656.7	23.644/4186.7	23.5513/1005.7	23.644/601.2
9	30/30	135.0137/17,414.6	138.3127/11,675.2	138.3988/16,887	138.0833/23,072.8	139.7357/25,470.7
10	40/40	144.0469/20,968.4	145.3967/21,086	146.4824/16,469.5	144.7231/25,026.3	144.6829/15,030.9

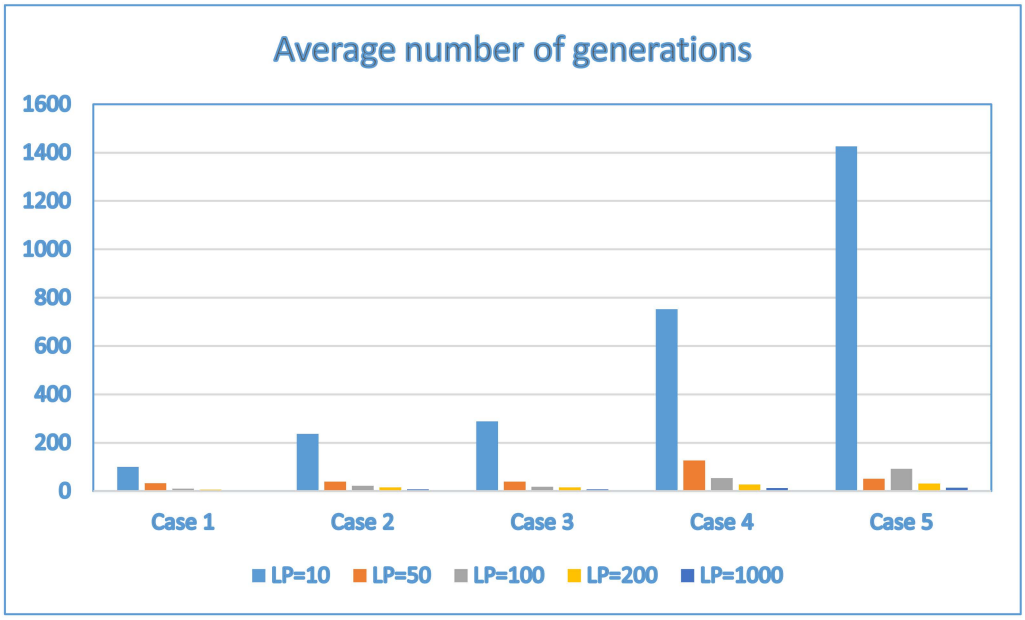

Figure 3. The average number of generations for Case 1, Case 2, Case 3, Case 4 and Case 5.

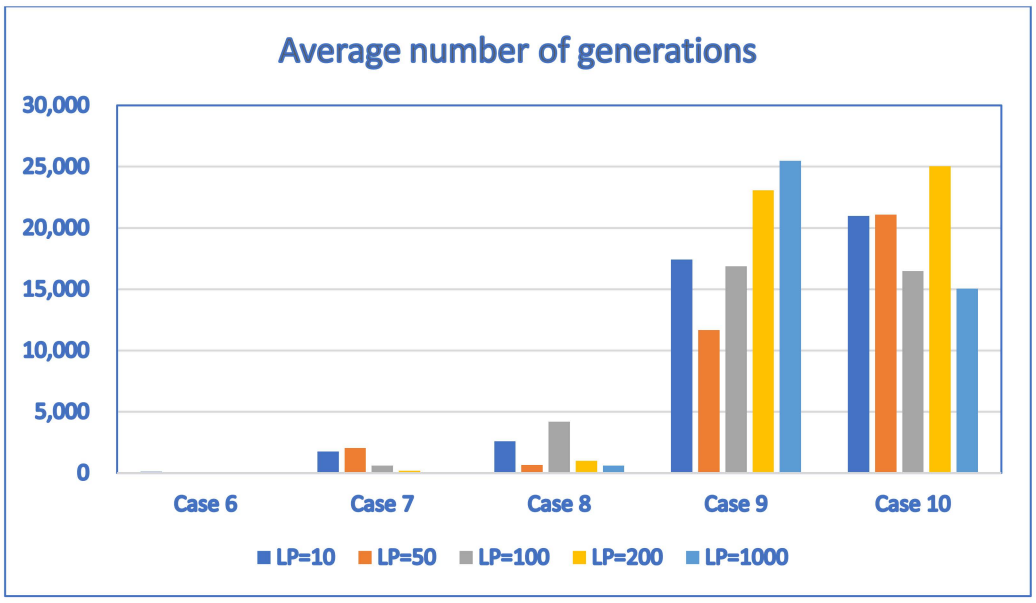

Figure 4. The average number of generations for Case 6, Case 7, Case 8, Case 9 and Case 10.

The effectiveness of an evolutionary algorithm is assessed based on two indices: the average fitness value and the average number of generations needed to find the solutions. The average fitness value is the primary index to evaluate performance, whereas the average number of generations is the secondary index to evaluate computational efficiency. Although the average number of generations for Case 9 with LP = 1000 is the worst, the average fitness value for Case 9 with LP = 1000 is 139.7357, which is the best of all, according to Table 17. Therefore, for Case 9, the SaNSDE algorithm with LP = 1000 still outperforms other LP parameters, in the average fitness value Table 17. The average number of generations for Case 9 with LP = 1000 is the worst. This is due to the fact that the SaNSDE algorithm is a stochastic optimization method. The number of generations to find the best solutions may not always be the smallest, even if it has learned the best strategy during the learning period. Despite the results of Case 9, the SaNSDE algorithm with LP = 1000 outperforms other LP parameters in all the other Cases, in terms of the

average fitness value and the average number of generations. This means that the SaNSDE algorithm, with a sufficient large learning period, has a higher probability to achieve better performance and efficiency.

5.3. Effectiveness of the Proposed Algorithm

In this subsection, we will compare the proposed algorithm with several evolutionary algorithms through the results of the experiments. These evolutionary algorithms include standard DE [14], NSDE [15], PSO [13], Firefly [26] and ALPSO [33] algorithms. Comparison with the standard DE algorithm will be presented first, followed by comparisons with NSDE, PSO, Firefly and ALPSO algorithms.

5.3.1. Comparison with DE

In this subsection, we compare the proposed algorithm with the standard DE algorithm [14]. As there are six mutation strategies in the standard DE approach, we use DE1, DE2, DE3, DE4, DE5 and DE6 to represent the six standard DE algorithms with the six mutation strategies.

The parameters used in the DE1, DE2, DE3, DE4, DE5 and DE6 are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
F: a value arbitrarily selected from uniform (0, 2).
$V_{\max}$ = 4.

The parameters used in the Self-adaptive Neighborhood Search Differential Evolution (SaNSDE) algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
$F_i = 0.3r_1 + 0.5$, where r_1 is a Gaussian distribution $N(0, 1)$.
LP = 1000.
$V_{\max}$ = 4.

Table 18 shows the results obtained by applying the DE algorithm with six strategies and the SaNSDE algorithm with algorithmic parameter LP set to 1000. The results indicate that the average fitness function values achieved by applying the DE algorithms are worse than those of the SaNSDE algorithm for some cases. In addition, the average fitness function values achieved by the SaNSDE algorithm are either as good as or better than DE. This indicates that the SaNSDE algorithm can stably find better solutions. In terms of convergence speed, the average number of generations required for the SaNSDE algorithm to find the best solutions is significantly smaller than those of the DE algorithm with six strategies, for most test cases, according to Table 18, Figures 5 and 6.

Table 18. Average fitness values and average number of generations for DE with six strategies and SaNSDE with algorithmic parameter $LP = 1000$.

Test Case	Participant (D/P)	DE1	DE2	DE3	DE4	DE5	DE6	SaNSDE ($LP = 1000$)
1	4/10	18.305/14.3	18.305/20.9	18.305/19.2	18.305/35.9	18.305/15.7	18.305/16	18.305/2.8
2	5/11	23.518/54.6	21.1662/129.1	22.7885/514.3	23.518/577.1	23.518/25.9	23.518/63.5	23.518/7.6
3	5/12	24.79/63.3	24.79/334.9	24.3962/69.6	24.3962/171.8	24.0024/413.2	24.3962/77.9	24.79/7.1
4	6/12	36.58/167.6	35.4095/419.5	36.58/214.2	35.9755/212.9	34.1465/202	36.58/135.1	36.58/12.9
5	7/13	29.2442/968.5	25.482/73.4	29.0007/181	26.2379/139.1	26.8132/478.5	29.2442/203.7	30.063/14.7
6	8/14	62.18/48.7	62.18/42.5	62.18/51.8	60.7055/1056.4	62.18/341.2	62.18/41.7	62.18/7
7	9/15	75.1808/860	70.481/165.5	78.1356/932.7	76.2821/1139.3	59.7039/1054.7	74.4286/1004	79.64/45.4
8	20/20	23.644/475	15.4555/9261.1	23.644/1098.6	16.1718/5027	16.0075/14,904.6	19.3717/6478.8	23.644/601.2
9	30/30	126.2855/26,507.9	81.1175/211.6	133.3518/20,689.3	108.7805/19,178	107.5981/15,306.9	114.6592/15,030	139.7357/25,470.7
10	40/40	120.0865/7556.1	79.7772/21,756.2	122.8838/22,660.2	84.8821/18,684.8	53.8039/18,172.4	55.685/12,640.3	144.6829/15,030.9

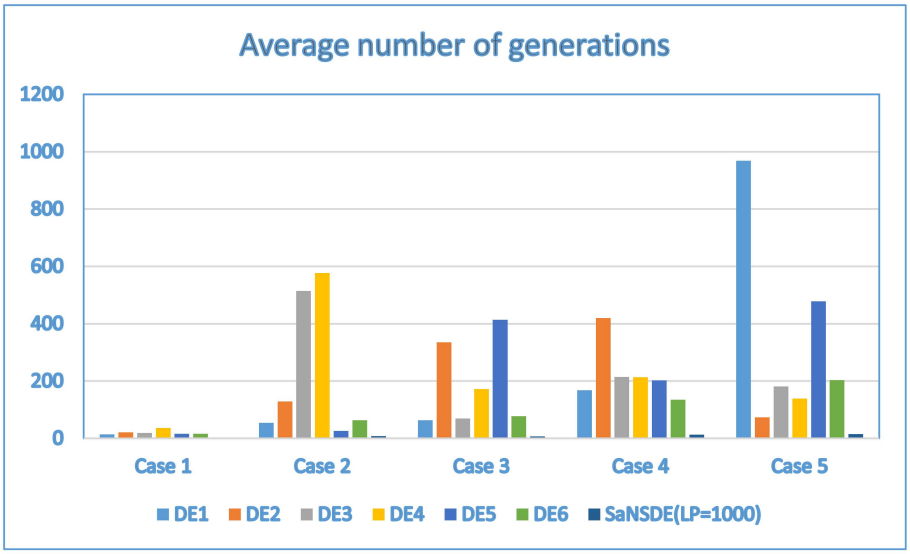

Figure 5. The average number of generations obtained by DE1, DE2, DE3, DE4, DE5, DE6 and SaNSDE with $LP = 1000$ for Case 1, Case 2, Case 3, Case 4 and Case 5.

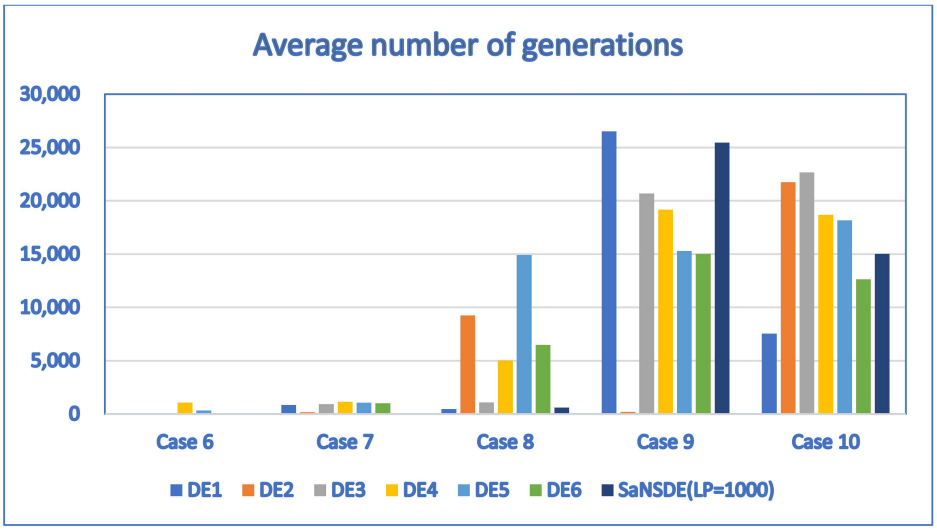

Figure 6. The average number of generations obtained by DE1, DE2, DE3, DE4, DE5, DE6 and SaNSDE with $LP = 1000$ for Case 6, Case 7, Case 8, Case 9 and Case 10.

5.3.2. Comparison with NSDE

In this subsection, we compare the proposed algorithm with the NSDE algorithm [15].

The parameters used in the Neighborhood Search Differential Evolution algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
$F_i = 0.5r_1 + 0.5$, where r_1 is a Gaussian distribution $N(0,1)$.
$V_{\max} = 4$.

The parameters used in the Self-adaptive Neighborhood Search Differential Evolution (SaNSDE) algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
$F_i = 0.3r_1 + 0.5$, where r_1 is a Gaussian distribution $N(0,1)$.
LP = 1000.
$V_{\max} = 4$.

Table 19 shows the results obtained by applying the NSDE algorithm and the SaNSDE algorithm, with algorithmic parameter LP set to 1000. The results indicate that the average fitness function values achieved by the SaNSDE algorithm are either as good as or better than the NSDE algorithm for all test cases. In addition, the average number of generations required for the SaNSDE algorithm to find the best solutions is significantly smaller than those of the NSDE algorithm, for all test cases, according to Table 19, Figures 7 and 8. In short, the SaNSDE algorithm is more efficient than the NSDE algorithm.

Table 19. Average fitness values and average number of generations for NSDE and SaNSDE with algorithmic parameter LP = 1000.

Test Case	Participant (D/P)	NSDE	SaNSDE (LP = 1000)
1	3/10	18.305/106.9	18.305/2.8
2	5/11	23.518/51.2	23.518/7.6
3	5/12	24.79/49.8	24.79/7.1
4	6/12	36.58/154.54	36.58/12.9
5	7/13	30.063/118	30.063/14.7
6	8/14	62.18/36.9	62.18/7
7	9/15	79.64/2173.4	79.64/45.4
8	20/20	23.5513/911.6	23.644/601.2
9	30/30	137.655/33,899.3	139.7357/25,470.7
10	40/40	137.2235/29,677.1	144.6829/15,030.9

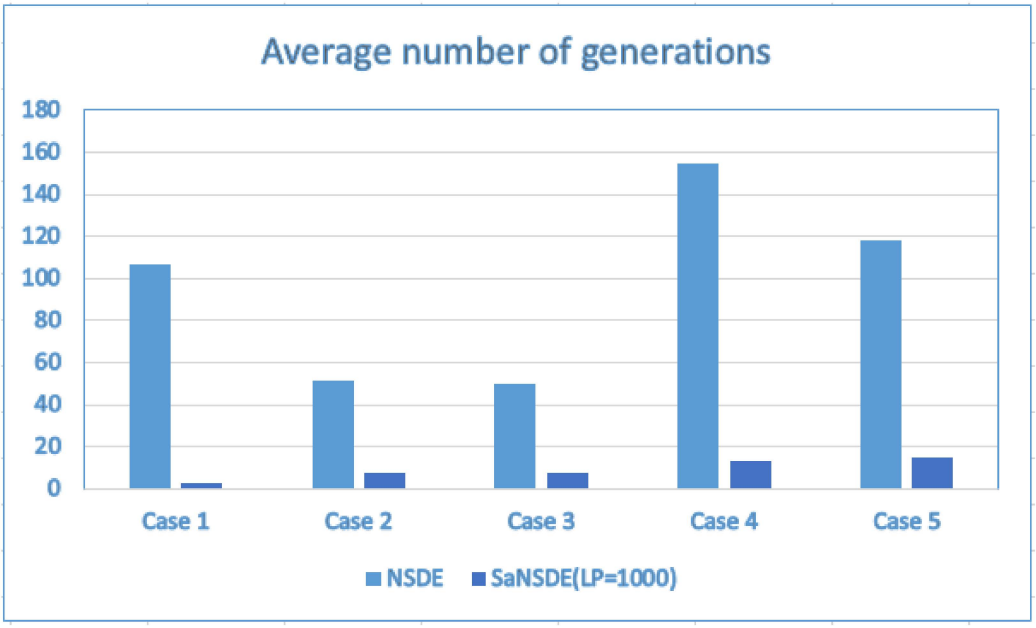

Figure 7. The average number of generations obtained by NSDE and SaNSDE with $LP = 1000$ for Case 1, Case 2, Case 3, Case 4 and Case 5.

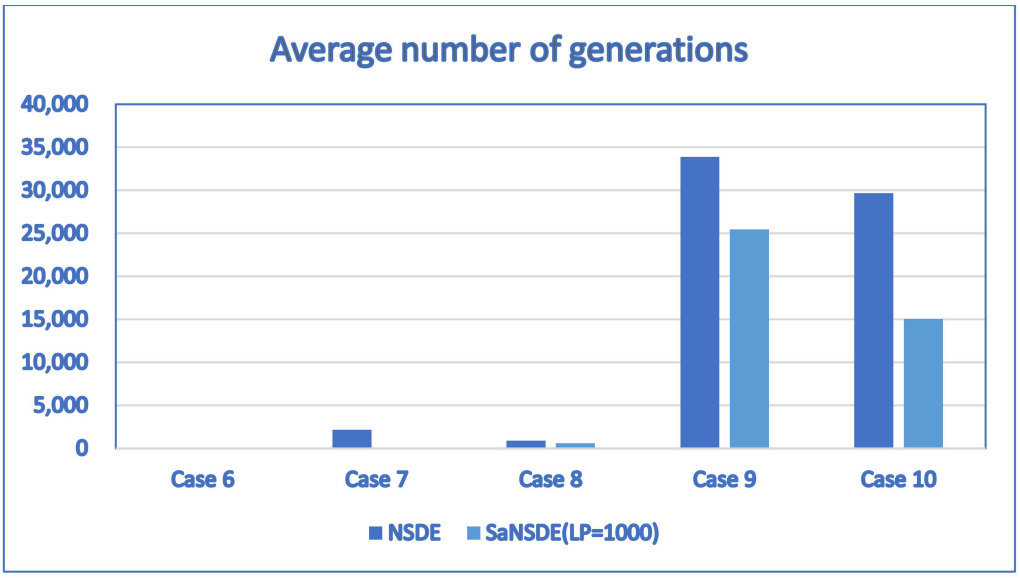

Figure 8. The average number of generations obtained by NSDE and SaNSDE with $LP = 1000$ for Case 6, Case 7, Case 8, Case 9 and Case 10.

5.3.3. Comparison with PSO, FA and ALPSO

In this subsection, we compare the proposed algorithm with the PSO algorithm [13], Firefly algorithm [26] and ALPSO algorithm [33].

The parameters used in the PSO algorithm are listed as follows:

$MAX\_GEN = 10{,}000$.
Population size $NP = 30$.
$c_1 = 0.4$.
$c_2 = 0.6$.
$\omega = 0.4$.
$V_{\max} = 4$.

The parameters used in the Firefly algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
β_0 = 1.0.
γ = 0.2.
α = 0.2.
$V_{\max}$ = 4.

The parameters used in the ALPSO algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
c_1 = 0.4.
c_2 = 0.6.
ω = 0.4.
$V_{\max}$ = 4.

The parameters used in the Self-adaptive Neighborhood Search Differential Evolution (SaNSDE) algorithm are listed as follows:

MAX_GEN = 10,000.
Population size NP = 30.
CR = 0.5.
$F_i = 0.3r_1 + 0.5$, where r_1 is a Gaussian distribution $N(0, 1)$.
LP = 1000.
$V_{\max}$ = 4.

Table 20 shows the results obtained by applying the PSO algorithm, Firefly algorithm (FA), ALPSO algorithm and the SaNSDE algorithm, with algorithmic parameter LP set to 1000. The results indicate that the average fitness function values achieved by applying the PSO algorithm, Firefly algorithm (FA) and ALPSO algorithm are worse than those of the SaNSDE algorithm, for Test Case 8 through Test Case 10. This indicates that the SaNSDE algorithm can stably find better solutions. In terms of convergence speed, the average number of generations required for the SaNSDE algorithm to find the best solutions is significantly smaller than those of the PSO algorithm, Firefly algorithm (FA) and ALPSO algorithm, for most test cases, according to Table 20, Figures 9 and 10. Therefore, the SaNSDE algorithm also enjoys faster convergence speed.

Table 20. Average fitness values and average number of generations for PSO, FA and SaNSDE with algorithmic parameter LP = 1000.

Test Case	Participant (D/P)	PSO	FA	ALPSO	SaNSDE (LP = 1000)
1	3/10	18.305/43.6	18.305/52.7	18.305/82.5	18.305/2.8
2	5/11	23.518/233.1	20.9662/134.9	23.518/266.6	23.518/7.6
3	5/12	24.79/691.8	24.3962/328.3	24.79/406.9	24.79/7.1
4	6/12	36.58/929.1	36.58/573.1	36.58/548.6	36.58/12.9
5	7/13	30.063/969.5	30.063/1618.6	30.063/1196.6	30.063/14.7
6	8/14	62.18/638.3	62.18/546.5	62.18/483.4	62.18/7
7	9/15	79.64/2643.4	77.3834/3206	79.64/1245.1	79.64/45.4
8	20/20	15.6939/27,714.3	10.8847/19,374.3	15.7567/30,399.2	23.644/601.2
9	30/30	27.9224/28,346.9	−2.3761/23,731.4	28.9539/32,209	139.7357/25,470.7
10	40/40	−2.5351/29,242.7	−4.0021/21,267.9	−0.5502/29,617.2	144.6829/15,030.9

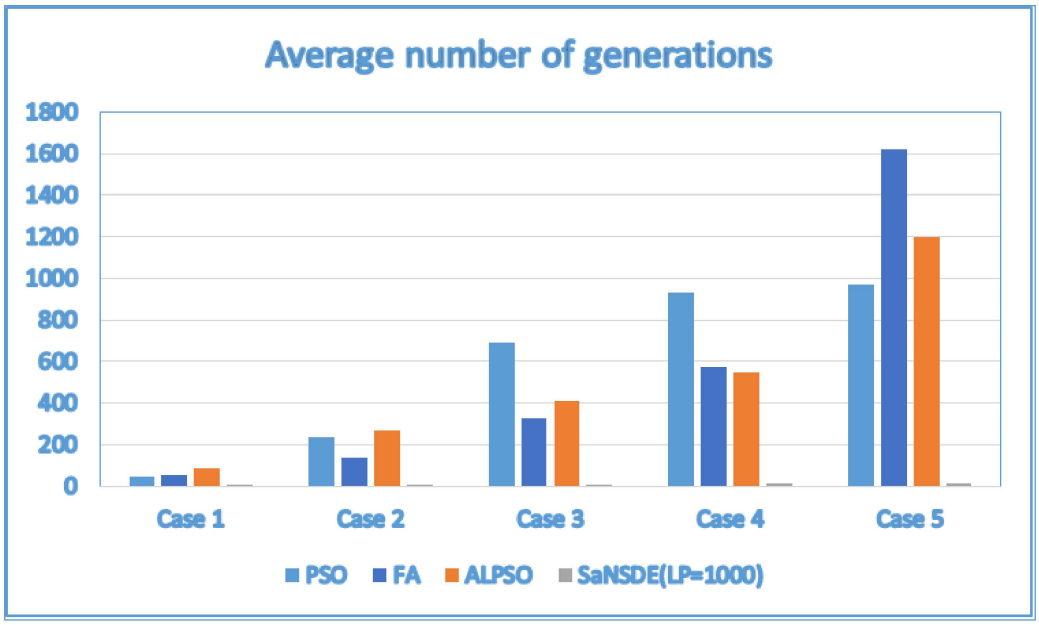

Figure 9. The average number of generations obtained by PSO, FA, ALPSO and SaNSDE with $LP = 1000$ for Case 1, Case 2, Case 3, Case 4 and Case 5.

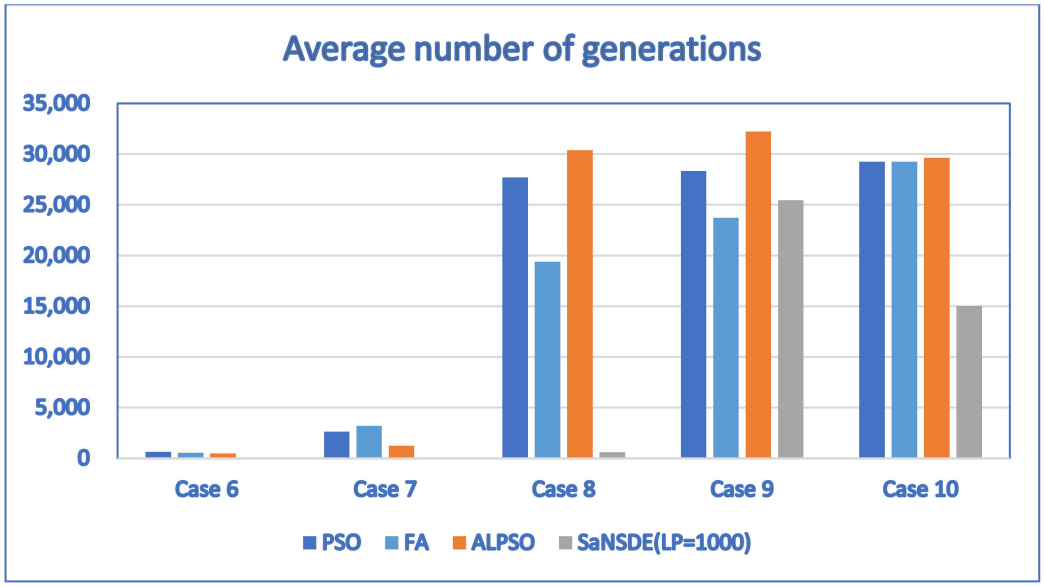

Figure 10. The average number of generations obtained by PSO, FA, ALPSO and SaNSDE with $LP = 1000$ for Case 6, Case 7, Case 8, Case 9 and Case 10.

The experiments are conducted on a laptop with Intel$^{(R)}$ Core$^{(TM)}$ i7 CPU, 16 GB of memory on board and a base clock speed of 2.6 GHz. The average CPU time for SaNSDE with $LP = 1000$ is shown in Figure 11. Case 9 takes about (over) 3 min. Case 10 takes about (over) 5 min. The CPU time can be cut down if a newer laptop with Intel$^{(R)}$ Core$^{(TM)}$ i9 CPU with a base clock speed of 3.5 GHz and a turbo clock speed of 4.8 GHz is used.

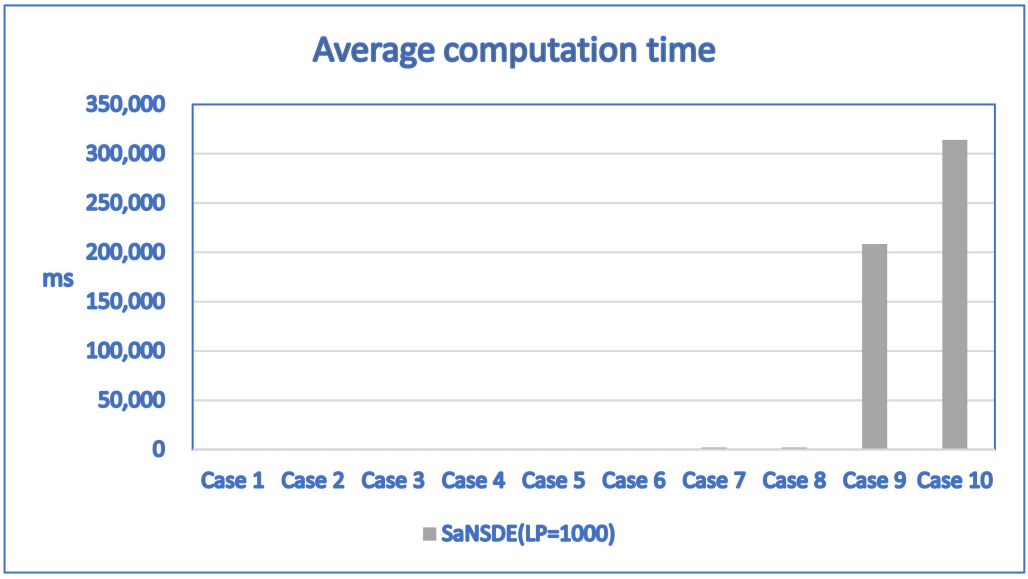

Figure 11. The average CPU time of SaNSDE with $LP = 1000$ for all Cases.

6. Discussion

Ridesharing has long been recognized as one potential way to improve sustainability in cities, by reducing energy consumption and greenhouse gas emissions. Besides the factors of cost and time, due to the relatively high rideshare crime rate, safety and trust become major determinants for the adoption of ridesharing. This paper aims to propose a social network assisted decision-making formulation for trust-based ridesharing and a more effective self-adaptive neighborhood search algorithm for solving the trust-based ridesharing problem. The proposed SaNSDE algorithm searches solutions by jointly taking into account scale factor, neighborhood search concept and self-adaptation of mutation strategies in the stochastic optimization process.

The lack of considering the trust factor is one of the main obstacles for people to adopt the ridesharing mode of transport in daily life. In this study, we propose a decision model to take the trust requirements of drivers and riders into account in the ridesharing problem. In this paper, a shared mobility system that considers the trust requirements of passengers and drivers is called a trust-based shared mobility system. To consider the trust factor in the ridesharing problem, it is assumed that a driver and a passenger can share a ride only if their trust requirements can be satisfied. A directed graph-based social network model is adopted in this study. Trust between drivers and passengers is described based on their social network. The trust level between nodes is represented by weight, associated with the set of edges in the social network model. The trust requirements are related to the trust level between drivers and passengers in the social network. In a trust-based shared mobility systems, a driver may request his/her minimal trust level requirements to share a ride with a passenger. If minimal trust level requirements requested by the driver cannot be satisfied, the driver will not share a ride with the passenger. Similarly, a passenger may request his/her minimal trust level requirements to share a ride with a driver. If minimal trust level requirements requested by the passenger cannot be satisfied, the passenger will not share a ride with the driver. Based on the trust level specified in the social network model, and minimal trust level requirements requested by the drivers/passengers, an optimization decision problem is formulated to find the solution for ridesharing recommendations. Due to computational complexity, a metaheuristic algorithm based on the Differential Evolution approach has been developed to solve the trust-based ridesharing decision problem and assess its effectiveness.

We verified the functionality to meet the trust requirements of drivers and passengers and studied the characteristics and effectiveness of the proposed algorithm. To verify the functionality to satisfy the trust requirements of drivers and passengers, we applied the

proposed algorithm to two examples of the trust-based ridesharing problem and analyzed based on the outputs of the algorithm. The results indicate the proposed algorithm can meet the trust requirements of drivers and passengers.

To study the characteristics of the proposed algorithm, we conducted several experiments by changing the learning period parameter. The results indicate that the average number of generations required to find the best solutions tends to decrease by increasing LP for most test cases. This is due to the fact that the best strategy can be found only if the learning period parameter LP is big enough. If the algorithmic parameter LP is too small, the best strategy cannot be learned within the learning period. In this case, the probability to select the best strategy to create a mutant vector will be lower than expected and, therefore, the number of generations required to find the best solutions will be greater. If the algorithmic parameter LP is big enough, the best strategy can be learned within the learning period. In this case, the probability to select the best strategy to create a mutant vector will be higher and, hence, the number of generations required to find the best solutions will be smaller.

To study the effectiveness of the proposed algorithm, we performed a number of experiments on several test cases by applying the proposed SaNSDE algorithm and other algorithms, including DE, NSDE, PSO, FA and ALPSO algorithms, and analyzed the numerical results of the experiments. The results indicate that the average fitness function values achieved by the SaNSDE algorithm are either as good as or better than DE, NSDE, PSO, FA and ALPSO. In terms of convergence speed, the average number of generations required for the SaNSDE algorithm to find the best solutions is significantly smaller than those of the DE, NSDE, PSO, FA and ALPSO algorithms for most test cases. Therefore, the SaNSDE algorithm enjoys faster convergence speed.

The effectiveness of the SaNSDE algorithm is due to the mechanism to learn better strategies in the optimization. After the learning period, the SaNSDE algorithm will select better strategies in the evolution processes. The number of generations required for many evolutionary algorithms to find the best solutions grows dramatically with the problem scale. Therefore, the number of generations required by the SaNSDE algorithm to find the best solutions will also increases dramatically. The advantage is that the SaNSDE algorithm is able to identity better strategies to find better solutions and jointly switch between strategies. The issue to deal with larger cases is an interesting future research direction.

There are alternatives to deal with passengers and drivers that may enter the ridesharing system sequentially in the real world. Typically, a decision support system for ridesharing makes decisions within a time period. The requests of drivers and passengers arriving before the start of the decision time period will be considered by the ridesharing decision support system. The decision support system will determine the matched drivers and passengers. Unmatched drivers and passengers will be considered in the next time period. The decision model proposed in this paper supports multi-drivers/multi-passengers, multi-drivers/single-passenger and single-driver/multi-passengers. Although the decision model proposed in this paper is based on the multi-driver/multi-passenger scenario, it can be applied to multi-driver/single-passenger and single-driver/multi-passenger scenarios. The proposed decision support system for ridesharing may consider each passenger and the potential drivers as a matching problem and solves the problem one after another. It may also consider each driver and the potential passengers as a matching problem and solves the problem one by one. There is a trade-off between performance and response time of different application scenarios, mentioned above. A ridesharing decision support system based on the multi-driver/multi-passenger scenario may achieve better performance but need more computational time. A ridesharing decision support system based on the multi-driver/multi-passenger scenario may achieve better performance but may not be able to respond to users quickly, as it requires more computational time. A ridesharing decision support system based on the multi-driver/single-passenger or single-driver/multi-passenger scenarios need less computational time and provide quick responses to users, but the performance may be degraded.

7. Conclusions

Due to the relatively high rideshare crime rate, safety and trust become important factors for the adoption of ridesharing, besides the factors of cost and time. One way to ensure safety/trust is to take advantage of the social relationship of ridesharing participants in relevant social networks. Although trust level may be linked to social distance, it is not the same as social distance. A proper model to capture the level that one participant trusts another is to consider both the social distance and other factors not directly relevant to social distance. In this study, we consider a trust model that takes into account social distance as well as non-social distance factors. In this paper, we have taken the first step to develop a well-compatible ridesharing decision-making algorithm, taking into account the trust factor in the poly-variant stochastic optimization process. This study aims to develop a decision model and associated solution methods to ensure the satisfaction of trust requirements of ridesharing participants. In the proposed decision model, the trust level between nodes is represented by weight, associated with the set of edges in the social network model. In a trust-based shared mobility system, a driver may request his/her minimal trust level requirements to share a ride with a passenger. A ride can be shared between a driver and relevant passengers only if the minimal trust level requirements requested by the driver and relevant passengers are satisfied. Based on the trust level specified in the social network model and minimal trust level requirements requested by the drivers/passengers, an optimization problem is formulated to find the solution. Due to computational complexity, a self-adaptive DE with neighborhood search (SaNSDE) algorithm, based on the Differential Evolution approach, has been developed to solve the trust-based ridesharing decision problem and assess its effectiveness.

To study the effectiveness of the proposed algorithm, we performed a number of experiments by applying the proposed SaNSDE, DE, NSDE, PSO, FA and ALPSO algorithms to find solutions for several test cases. The results indicate that the solutions found by the SaNSDE algorithm are either as good as or better than DE, NSDE, PSO, FA and ALPSO algorithms. In addition, the SaNSDE algorithm outperforms DE, NSDE, PSO, FA and ALPSO algorithms in terms of convergence speed for most test cases. This study shows that the SaNSDE algorithm can be applied to generate ridesharing recommendations effectively. One future research direction is to apply the SaNSDE algorithm to other types of recommender systems or applications.

Funding: This research was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 110-2410-H-324-001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository described in the article.

Acknowledgments: The author would like to thank the anonymous reviewers and editors for their comments and suggestions, which are invaluable for the author to improve the presentation, clarity and quality of this paper.

Conflicts of Interest: The author declares no conflict of interest.

References

1. McKinsey and Company. Shared Mobility: Where It Stands, Where It's Headed. 2021. Available online: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/shared-mobility-where-it-stands-where-its-headed (accessed on 10 January 2022).
2. Agatz, N.; Erera, A.; Savelsbergh, M.; Wang, X. Optimization for dynamic ride-sharing: A review. *Eur. J. Oper. Res.* **2012**, *223*, 295–303. [CrossRef]
3. Furuhata, M.; Dessouky, M.; Ordóñez, F.; Brunet, M.; Wang, X.; Koenig, S. Ridesharing: The state-of-the-art and future directions. *Transp. Res. Part B Methodol.* **2013**, *57*, 28–46. [CrossRef]
4. Mourad, A.; Puchinger, J.; Chu, C. A survey of models and algorithms for optimizing shared mobility. *Transp. Res. Part B Methodol.* **2019**, *123*, 323–346. [CrossRef]

5. Martins, L.C.; Torre, R.; Corlu, C.G.; Juan, A.A.; Masmoudi, M.A. Optimizing ride-sharing operations in smart sustainable cities: Challenges and the need for agile algorithms. *Comp. Ind. Eng.* **2021**, *153*, 107080. [CrossRef]
6. Tang, L.; Duan, Z.; Zhao, Y. Toward using social media to support ridesharing services: Challenges and opportunities. *Transp. Plan. Technol.* **2019**, *42*, 355–379. [CrossRef]
7. Bistaffa, F.; Farinelli, A.; Ramchurn, S.D. Sharing rides with friends: A coalition formation algorithm for ridesharing. In Proceedings of the 29th AAAI Conference on Artificial Intelligence Pattern, Austin, TX, USA, 25–30 January 2015; pp. 608–614.
8. Bistaffa, F.; Farinelli, A.; Chalkiadakis, G.; Ramchurn, S.D. A cooperative game-theoretic approach to the social ridesharing problem. *Artif. Intell.* **2017**, *246*, 86–117. [CrossRef]
9. Li, Y.; Chen, R.; Chen, L.; Xu, J. Towards Social-Aware Ridesharing Group Query Services. *IEEE Trans. Serv. Comp.* **2017**, *10*, 646–659. [CrossRef]
10. Fu, X.; Zhang, C.; Lu, H.; Xu, J. Efficient Matching of Offers and Requests in Social-Aware Ridesharing. In Proceedings of the 2018 19th IEEE International Conference on Mobile Data Management (MDM), Aalborg, Denmark, 25–28 June 2018; pp. 197–206.
11. Xia, J.; Curtin, K.M.; Huang, J.; Wu, D.; Xiu, W.; Huang, Z. A carpool matching model with both social and route networks. *Comp. Environ. Urban Syst.* **2017**, *75*, 90–102. [CrossRef]
12. Shim, C.; Sim, G.; Chung, Y.D. Cohesive Ridesharing Group Queries in Geo-Social Networks. *IEEE Access* **2020**, *8*, 97418–97436. [CrossRef]
13. Hsieh, F.S. A Particle Swarm Optimization Algorithm to Meet Trust Requirements in Ridesharing Systems. In Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 28–31 October 2020; pp. 592–595.
14. Hsieh, F.S. A Differential Evolution Approach to Trust based Ridesharing Systems. In Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, USA, 13–16 October 2021; pp. 1–6.
15. Hsieh, F.S. Neighborhood Search in Differential Evolution for Solving Trusted Ridesharing Problems. In Proceedings of the IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEEE IEMCON 2021), Vancouver, BC, Canada, 27–30 October 2021; pp. 46–51.
16. Kennedy, J.; Eberhart, R.C. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
17. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
18. Hsieh, F.S.; Zhan, F.; Guo, Y. A solution methodology for carpooling systems based on double auctions and cooperative coevolutionary particle swarms. *Appl. Intell.* **2019**, *49*, 741–763. [CrossRef]
19. Hsieh, F.S. A Comparative Study of Several Metaheuristic Algorithms to Optimize Monetary Incentive in Ridesharing Systems. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 590. [CrossRef]
20. Hsieh, F.-S. A Comparison of Three Ridesharing Cost Savings Allocation Schemes Based on the Number of Acceptable Shared Rides. *Energies* **2021**, *14*, 6931. [CrossRef]
21. Waerden, P.; Lem, A.; Schaefer, W. Investigation of Factors that Stimulate Car Drivers to Change from Car to Carpooling in City Center Oriented Work Trips. *Transp. Res. Proced.* **2015**, *10*, 335–344. [CrossRef]
22. Shaheen, S.A.; Chan, N.D.; Gaynor, T. Casual carpooling in the San Francisco Bay Area: Understanding user characteristics, behaviors, and motivations. *Transp. Policy* **2016**, *51*, 165–173. [CrossRef]
23. Bough, C. Rideshare Crime Rates and Safety Tips for Women: Understand the Risks of Your Trip. 2021. Available online: https://www.reviews.com/insurance/car/ride-share-safety-tips/ (accessed on 20 January 2022).
24. Meshram, A.; Choudhary, P.; Velaga, N.R. Assessing and Modelling Perceived Safety and Comfort of Women during Ridesharing. *Transp. Res. Proced.* **2020**, *48*, 2852–2869. [CrossRef]
25. Li, Y.; Wan, J.; Chen, R.; Xu, J.; Fu, X.; Gu, H.; Lv, P.; Xu, M. Top-*k*k Vehicle Matching in Social Ridesharing: A Price-Aware Approach. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 1251–1263.
26. Yang, X.S. Firefly algorithms for multimodal optimization. In *Stochastic Algorithms: Foundations and Applications*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5792, pp. 169–178.
27. Bilal; Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential Evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479. [CrossRef]
28. Zhang, Y.; Wang, S.; Ji, G. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Math. Probl. Eng.* **2015**, *2015*, 931256. [CrossRef]
29. Fister, I.; Fister, I., Jr.; Yang, X.-S.; Brest, J. A comprehensive review of firefly algorithms. *Swarm Evolut. Comput.* **2013**, *13*, 34–46. [CrossRef]
30. Yang, Z.; He, J.; Yao, X. Making a difference to differential evolution. In *Advances in Metaheuristics for Hard Optimization*; Michalewicz, Z., Siarry, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 415–432.
31. Qin, A.K.; Suganthan, P.N. Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 1785–1791.
32. Yang, Z.; Tang, K.; Yao, X. Self-adaptive differential evolution with neighborhood search. In Proceeding of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 1110–1116.

33. Wang, F.; Zhang, H.; Li, K.; Lin, Z.; Yang, J.; Shen, X.L. A hybrid particle swarm optimization algorithm using adaptive learning strategy. *Inf. Sci.* **2018**, *436–437*, 162–177. [CrossRef]
34. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]