Received November 14, 2019, accepted December 10, 2019, date of publication January 10, 2020, date of current version January 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965557

Smart City Response to Homelessness

PEDRAM KHAYYATKHOSHNEVIS[1], SALIMUR CHOUDHURY[1], ERIC LATIMER[2], AND VIJAY MAGO[1]
[1]Department of Computer Science, Lakehead University, Thunder Bay, ON P7B 5E1, Canada
[2]Department of Psychiatry, McGill University, Montrèal, QC H3A 0G4, Canada

Corresponding author: Vijay Mago (vmago@lakeheadu.ca)

This work was supported in part by the Insight through the Social Sciences and Humanities Research Council under Grant 21820.

ABSTRACT The smart city is a concept of utilizing digital technologies to improve and enhance the lives of a city's inhabitants. This concept has been the subject of increasing interest over the past few years. However, most studies address improving aspects of a city's infrastructure, such as information security, privacy, communication networks, government, and transportation. Noticeably absent from the subject matter of these studies are social problems, such as poverty and homelessness. In this paper, we explore how technology can be harnessed to mitigate homelessness. We introduce eight novel heuristic algorithms that create a desirable homeless-to-housing assignment with regards to homeless individuals' characteristics and the nature of services. We discuss the efficiency of each of the algorithms through simulations. Our best performing algorithm obtains 92% accuracy in comparison to the optimal solution and 99.7% fairness. The algorithms are compared in terms of execution time, solution accuracy, fairness, and the relative difference with the optimal solution of this NP-hard problem.

INDEX TERMS Smart cities, homelessness, social systems, greedy algorithm, local search algorithm.

I. INTRODUCTION

Urbanization is a migration phenomenon in which large populations migrate from rural areas to urban regions. Currently, half of the planet's population is living in urban areas. It is expected that, within five decades, seven out of ten humans will be living in urban areas [1]. Rapid urbanization is among the primary factors that cause the rise in the population experiencing homelessness [2]. Additionally, the 2008 economic crisis was followed by an economic downturn [3]. Homelessness has been one of the persistent and important issues across most developed nations. It has been estimated that at least 235,000 Canadians experience homelessness at some point over the course of a year [4]. These include disproportionate numbers of men, young adults who have gone through the youth protection system, Indigenous, and LGBTQ2S, among other subgroups. Homelessness, especially street homelessness, is associated with increased mortality [5]. Homelessness is also associated with many adverse outcomes, including substance abuse and justice system involvement [6]. In Canada's larger cities, homeless people with mental illness have been estimated to cost about $60,000 per person, per year in health, social, criminal justice and other services [7]. In total, homelessness has been conservatively estimated to cost the Canadian economy about $7.6 billion annually. The Canadian government has initiated several programs to provide decent and safe housing to homeless individuals. Homeless individuals may spend one or more nights in a variety of types of places, including street locations, emergency shelters, violence against women (VAW) shelters, and various forms of transitional housing (transitional housing typically includes on-site support staff and stays are limited to a maximum duration that ranges from a few months to as many as 5 years). Although current policy directions seek to reduce reliance on emergency shelters and transitional housing and quickly transition people who are homeless directly into permanent housing (often with support), emergency shelters and transitional housing remain essential components of the homelessness service system across Canada. On a given night, over 14,000 Canadians are estimated to find themselves in an emergency homelessness shelter. In Montreal, the number of people in transitional housing exceeds those in emergency shelters [4], [6]. Most shelters and housing providers have specific admission criteria that include factors such as gender, age group, sobriety, and Indigenous ancestry. Even for someone with a particular profile (e.g., a 35-year-old non-Indigenous man who can show up

The associate editor coordinating the review of this manuscript and approving it for publication was M. A. Hannan.

reliably in a non-intoxicated state, or a woman who has experienced abuse from her partner), there may be several available housing options in a city. The person will seek admission to one place and then to another if the first is full. Shelter personnel (or sometimes case managers) may help a person access transitional housing from an emergency shelter. However, when this happens, the process will invariably be based on limited information. The development of low-cost web platforms combined with high computational capability provides an opportunity to make this process much more efficient by reducing trial and error and more quickly matching individuals to housing providers that are likely to be better suited to their needs.

A. MOTIVATING SCENARIO

In this subsection, we provide a few examples to illustrate how individuals can benefit from such platforms. Consider the following situations (names have been changed to protect the privacy of individuals):

- Diane is 44 years old and has been homeless twice in her life. Once when she was 23 and again at 30. When she was 23, she was asked by her dad to leave her family house. She was struggling and went to stay with friends. Because of her panic attack, she was asked to leave their house. She was scared, confused and wandering in the street. She was eventually helped and got back on her feet. Her struggles could have been less if she was promptly assigned to the most suitable service provider.
- Olivia made an exhausting two-month journey to a different city. She is often dehydrated, has severe stomach pain, and no place to go. She can benefit from a centralized, easy to use application that can recommend the most appropriate housing provider to assist her with her needs.
- Haley was diagnosed with schizoaffective disorder and also experiences gender dysphoria. Haley's parents forced her out of her house. She is scared of going to a homeless shelter because of the negative comments she has heard about homophobic violence or abuse against LGBTQ2S individuals. Haley does not know this, but there are a few service providers nearby her that specialize in assisting LGBTQ2S homeless individuals.

These vignettes illustrate how a centralized mechanism that appropriately matches diverse homeless individuals to housing providers (taking into consideration factors like gender, age group, sobriety, Indigenous ancestry, and the distance to the provider) could improve the living experience of homeless individuals and play a significant role in their social reintegration. Information gathered from such a system (e.g., shortages of specific types of housing) would also be of great value to policy-makers, who currently often rely only on impressionistic and anecdotal information.

B. SUMMARY OF THE RESEARCH

Currently, the process by which homeless individuals are distributed among available emergency shelters and transitional housing is highly decentralized. In this study, we propose eight novel heuristic algorithms used to create a suitable one-to-one homeless-to-housing matching. Web or mobile applications using these algorithms can be developed to assist homeless individuals by providing them with options for choosing housing that best matches their needs and circumstances. In addition, implementation of such a system can be used to provide policymakers with helpful system-level data. The Conclusions and Future Work section provides a detailed use case of the Smart Housing Framework.

Throughout this study, we use *weight* as a metric to define the relationship between an individual and a particular housing provider, taking into account the Goodness-of-fit of the assignment. The *weight* is inversely proportional to the Goodness-of-fit of the relationship. The main objective of the proposed algorithms is to produce an assignment solution in which every homeless individual is matched with the most appropriate housing provider. Besides that, we are also interested in decreasing the computation time and increasing the *Fairness Index*[1] of the entire assignment solution. The *Fairness Index* is the collective equality of the Goodness-of-fit among all homeless individuals within the assignment solution. Therefore, we will compare our algorithms based on execution time, assignment's *weight*, and the *Fairness Index*. The accuracy of the final solution is evaluated by comparison to that of the optimal solution.

II. RELATED WORK

Minimizing the weight of the homeless-to-housing assignment is similar to minimizing the quantity that is known as makespan [8]. Makespan is a load balancing problem where m machines are given a set of n jobs. A set of jobs, a_i, are assigned to machine m_i. Machine m_i needs to work for a total time of m_t. This is declared as the load on machine m_i. Makespan is the maximum load on any machine in set M, and the scheduling problem by finding the assignment with minimum makespan is an NP-hard problem.[2] Similar to makespan, we are interested in an assignment solution where every homeless individual is offered a housing provider with minimum *weight* based on a set of constraints. Because of this similarity, we will summarize relevant makespan research in this section. Furthermore, we are interested in literature that embraces social science and software tools. The related work section of this research focuses on social studies, social systems, and optimal resource allocation.

[1]Jain's fairness index is a quantitative measure that is independent of the number of resources, or in this case the number of individuals or the range of weights. The fairness index represents the "equality" of the assignment's weight that is allocated to each individual. If all individuals get assignments with the same weights then the fairness index is 1, and the system is 100% fair. As the disparity increases, fairness decreases and a system that favors only a few individuals has a fairness index near 0.

[2]We use the term NP-hard referring to solving any NP-problem (non-deterministic polynomial time). NP-hard therefore, means "at least as hard as an NP-complete problem."

1) SOCIAL STUDIES AND SOCIAL SYSTEMS

Homelessness and poverty are complex problems affecting different types of individuals with different needs. Governments around the world have created various initiatives to address homelessness for different groups of people [9]. For instance, in 2013, the Canadian government budgeted C$119 million for new approaches that could partially solve homelessness [10]. The main one among these is Housing First, which provides homeless individuals, who so desire, their own subsidized, permanent housing, as well as individualized supports. While current policy direction favours expanding the availability of Housing First programs (to which our algorithms could assign a person), in practice emergency shelters, transitional housing and other types of housing (e.g., substance abuse treatment centers), are likely to remain an essential part of available services for years to come. The smart city is a concept that aims to help the city deliver services to citizens more efficiently. The smart city movement is mainly centered around information technology and innovation in government. Governments in smart cities mostly focus on projects with broad applications such as advancement in information technology, big data, and electronic delivery of information. However, less thought is given to how technology and data can solve social problems and help to eliminate intractable problems such as homelessness and poverty [9]. According to Kumar and Dahiya [11], emerging patterns of urbanization show that different countries require different policies, approaches, and strategies to create a prosperous smart city. Marvin *et al.* [12] surveys the economic and political aspects of smart cities. They also conduct a critical analysis of information security in smart cities, such as a possible breach of sensitive data. Neirotti *et al.* [13] studies the global definition of smart cities and lays out six main domains of research to shape future smart city strategies for policymakers.

2) OPTIMAL RESOURCE ALLOCATION

The growing interest in Internet of things (IoT) and the ever-increasing number of users provide expanding opportunities for applying optimization methods to resource allocation [14]. Improvements obtained from optimal resource allocation include energy usage reduction of sensors with limited energy, maximizing bandwidth, maximizing and maintaining the quality of service (QoS), minimizing communication collusion in network systems, and many other desirable functionalities and improvements, which results in minimizing cost and maximizing productivity. The resource allocation problem in computer science is similar to the problem statement presented in this paper. Therefore, we are interested in studies that address resource allocation problems. In the optimization field, makespan is the maximum load on any machine within a set of machines. Makespan is one of the most critical performance indicators, which has been the center of interest for many years [15]. Assignment with minimum makespan is obtained through optimal resource allocation, waste time reduction, and other methods that result in less energy usage, cost reduction, and productivity advancement [15]. Flow shop scheduling and network models are methods used to minimize the maximum load on a machine and maximize productivity [16], [17]. However, an essential factor is the trade-off between minimizing the maximum load and reliability or accuracy. Bi-objective algorithms can eliminate that trade-off using different optimization methods, such as wind driven optimization (WDO) or particle swarm optimization (PSO) [17], [18]. Heuristic algorithms that minimize the maximum load on a machine can be targeted to specific situations to fulfill a requirement, while still maintaining minimum task completion duration, such as preventive maintenance schedules or parallel machines with limited availability [19], [20]. These heuristic algorithms usually applied to NP-hard problems that require an instant solution.
Furthermore, an accurate makespan estimation is crucial. Makespan estimations are usually used as an indication of task completion, after which another task begins. Different Machine Learning approaches have been used to increase the makespan estimation accuracy, such as multilayer perceptron type Neural Network [15], which has proven using simulation to be superior to other methods, such as extreme learning machines, and Support vector machine algorithms, in terms of the regression performance indicator. However, a recent publication by the same authors produces better results using convolutional neural networks [21]. In the next section, we formulate the homeless-to-housing problem and define the objective of this research.

III. SYSTEM MODEL AND PROBLEM FORMULATION

This paper considers a system with m homeless individuals and n housing providers, where the set of homeless individuals is represented as $H = \{h_1, h_2, \ldots\ldots, h_m\}$ and the set of housing providers is represented as $S = \{s_1, s_2, \ldots\ldots, s_n\}$. From this point forward, homeless individuals are denoted as h_i, $1 \leq i \leq m$ and housing providers are denoted as s_j, $1 \leq j \leq n$. In this work, the assignment of h_i to s_j is denoted as $a_{i,j}$. This can be formally defined as:

$$a_{i,j} \in \{0, 1\}; \quad \forall i, j.$$
$$a_{i,j} = \begin{cases} 1, & \text{if } h_i \text{ is assigned to } s_j, \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Furthermore, a homeless individual can be assigned to only one housing provider. This constraint is defined as:

$$\sum_{j=1}^{n} a_{i,j} = 1; \quad \forall i \tag{2}$$

The capacity is the maximum number of h_i that can be assigned to s_j. Let c_j be the capacity of housing provider s_j. The capacity constraint is define as:

$$\sum_{i=1}^{m} a_{i,j} \leq c_j; \quad \forall j \tag{3}$$

The weight set W contains a relationship weight for every homeless individual and a housing provider. If $a_{i,j}$ denotes the assignment of the individual h_i to the housing provider s_j, then, $w_{i,j}$ represents the Goodness-of-fit of $a_{i,j}$ as an integer between 0 and 100. A lower $w_{i,j}$ value denotes a better matching between an individual and a housing provider. Our algorithms are programmed to utilize the given weights in order to create a better homeless-to-housing matching assignment. The accurate determination of each assignment weight with regards to the impact it has on each individual is a complex social problem and needs further in-depth research. High accuracy of this measurement will affect how well our algorithms impact social settings. However, since our algorithms are designed to utilize the given weights regardless of how they were decided we will randomly generate the weights for all the combinations of homeless individuals and housing providers.

$$w_{i,j} \geq 0; \quad \forall i, j. \tag{4}$$

Furthermore, set W can be presented as:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & w_{m,3} & \cdots & w_{m,n} \end{bmatrix}$$

The given problem consists of a set H (homeless individuals) and a set S (housing providers), where the matching process is subject to the capacity constraint. We approach this problem by introducing a combination of Greedy and local search algorithms. The algorithms attempt to solve the problem by performing greedy methods so that the final solution contains the lowest weight for each homeless individual without checking every possible pair of associated homeless clients and shelters. We refer to this objective as minimizing the maximum weight. It is important to distinguish between minimizing the sum of weights of the final solution and minimizing the maximum weight of the final solution. To better explain this approach, let us assume the following scenario: Given $H = \{h_1, h_2\}$, $S = \{s_1, s_2\}$, $C = \{1, 1\}$, and $W = \{[6, 1], [10, 6]\}$. The optimal solution for minimizing the sum of all weights is $A = \{a_{1,2}, a_{2,1}\}$ ($h_1 \rightarrow s_2, h_2 \rightarrow s_1$). The sum of weights of this assignment set is 11. However, the *maximum assigned weight* in that assignment set was 10 ($h_2 \rightarrow s_1$), which means h_1 was given a much better housing provider at the expense of h_2. This assignment set is considered to be an unfair solution since weights are distributed unevenly amongst individuals, or there is a large inequality between homeless individuals. A homeless-to-housing assignment is considered to be fair when the final solution has high equality of weights among all homeless individuals. Therefore, the optimal solution for minimizing the maximum weight is $A = \{a_{1,1}, a_{2,2}\}$, where the maximum weight is noticeably reduced compared to the previous solution. This objective can be formulated as:

$$min(max\ w_{i,j}.a_{i,j}) \tag{5}$$

The homeless-to-housing assignment is a Combination Optimization Problem (COP), in which the ideal outcome is reached by trying all solutions in the search space to find the best possible arrangement. Such problems are also known as NP-hard problems [22]. We formulate the introduced problem as:

$$\text{objective: } min(max\ w_{i,j}.a_{i,j}) \tag{6a}$$

$$\text{subject to: } \sum_{j=1}^{n} a_{i,j} = 1; \quad \forall i \tag{6b}$$

$$\sum_{i=1}^{m} a_{i,j} \geq 0; \quad \forall j. \tag{6c}$$

$$\sum_{i=1}^{m} a_{i,j} \leq c_j; \quad \forall j \tag{6d}$$

$$\text{where: } w_{i,j} \geq 0; \quad \forall i, j. \tag{6e}$$

$$a_{i,j} \in \{0, 1\}; \quad \forall i, j. \tag{6f}$$

The most straightforward method to find the optimal solution is to try the association of each homeless individual and housing provider. This method is referred to as the brute force method, and it is computationally expensive and time-consuming. In some cases, with a realistic population set, the brute force method takes hours to find the optimal solution. To further expand on this issue, let us consider a scenario with five homeless individuals, five housing providers, and a sum of five capacities in all the housing providers. For this problem, there are 120 unique combination sets. However, the number of combinations grows exponentially as the number of individuals and housing providers increases, or when the housing provider capacity decreases. For example, for 15 individuals, 15 housing providers, and 15 capacity among these housing providers, there are 1,307,674,368,000 unique combinations. Hence, this method is not practical given a large number of homeless individuals in major cities like Toronto, or cities with a smaller population like Montreal, with about 3,000 homeless people on a given night [6].

IV. PROPOSED ALGORITHMS

In this section, eight novel heuristic algorithms are proposed to solve the problem defined in (6b) to (6f) in polynomial time complexity. The first four algorithms are greedy algorithms with minor variations. The other four algorithms are local search algorithms that attempt to improve the solution output of the greedy algorithms.

A. GREEDY METHOD

The greedy algorithms loop through the homeless individuals set H and give priority to an individual with maximum weight values (algorithms are attempting to minimize the final assignment weights). The prioritized individual is sheltered promptly and then removed from set H. This process continues until either set H is empty or all the housing providers are at full capacity. The key contribution of this research is the following greedy algorithms:

1) STDEV Algorithm
2) Median Algorithm
3) Minmax Algorithm
4) Average Algorithm

Generally, the greedy algorithms prioritize a homeless individual that has a maximum disparity among its relationship weights (weight disparities are calculated differently by each algorithm). A maximum disparity suggests a more significant distance between the weights of best and the worst housing provider for that individual. Therefore, it is preferable to match that individual with a housing provider which carries the lowest weight before that housing provider is at full capacity. Consider the following scenario, $H = \{h_1, h_2\}$, $S = \{s_1, s_2\}$, $C = \{1, 1\}$, and $W = \{[2, 10], [3, 4]\}$. In this case, individual h_1 has a higher weight disparity comparing to individual h_2, and the capacity constraint creates a consequential decision, so that, if h_2 is assigned to s_1, then, h_1 must be assigned to s_2, which is the non-optimal solution. *Algorithm 1 : Greedy − Part 1* is explained as follows:

Algorithm 1 Greedy - Part 1

```
Input: $W : [[w_{1,1}, w_{1,2}, \dots, w_{1,n}],$
        $[w_{2,1}, w_{2,2}, \dots, w_{2,n}], [w_{m,1}, w_{m,2}, \dots, w_{m,n}]];$
    $H : [h_1, h_2, \dots, h_m];$
    $S : [s_1, s_2, \dots, s_n];$
    $C : [c_1, c_2, \dots, c_n];$
Output: $A$;
1  $A \leftarrow \emptyset$;
2  $Algo \leftarrow$ select one of the greedy algorithms ;
   // STDEV_Algorithm, Median_Algorithm,
   Minmax_Algorithm, or Average_Algorithm
3  while $H \neq \emptyset$ or $S \neq \emptyset$ do
4      $D \leftarrow Algo(W)$;
5      $personIndex \leftarrow$ get the index position of $max(D)$;
       // find the maximum value in $D$
6      $shelterIndex \leftarrow$ get the index position of
       $min(W[personIndex])$; // find the shelter
       with minimum weight for person
7      $a_{personIndex,\ shelterIndex} \leftarrow 1$; // assigne $h_i$ to
       $s_j$
8      $H \leftarrow H.remove(personIndex)$;
9      $W \leftarrow W.remove(personIndex)$;
10     $capacity \leftarrow C[shelterIndex]$;
11     $capacity \leftarrow capacity - 1$;
12     if $capacity == 0$ then
13         $S \leftarrow S.remove(shelterIndex)$;
14     end
15 end
16 return $A$
```

1) Line 1 to 3: Algorithm starts by initializing set A. In the next step, we choose one of the greedy algorithms. while-loop is initiated. The while-loop terminates when set H or set S is empty.

Algorithm 1 Greedy - Part 2

```
Input: $W : [[w_{1,1}, w_{1,2}, \dots, w_{1,n}],$
        $[w_{2,1}, w_{2,2}, \dots, w_{2,n}], [w_{m,1}, w_{m,2}, \dots, w_{m,n}]];$
Output: $D$
1  Function STDEV_Algorithm($W$):
2      $D \leftarrow \emptyset$;
3      for each row $i$ in $W$ do
4          $stdev \leftarrow$ get the standard deviation of row $i$;
5          $min \leftarrow$ get the minimum value of row $i$;
6          $D[i] \leftarrow stdev - min$;
7      end
8      return $D$
9  Function Median_Algorithm($W$):
10     $D \leftarrow \emptyset$;
11     for $i$ in $W$ do
12         $mdn \leftarrow$ get the median of row $i$;
13         $min \leftarrow$ get the minimum value of row $i$;
14         $D[i] \leftarrow mdn - min$;
15     end
16     return $D$
17 Function Minmax_Algorithm($W$):
18     $D \leftarrow \emptyset$;
19     for $i$ in $W$ do
20         $min \leftarrow$ get the minimum value of row $i$;
21         $max \leftarrow$ get the maximum value of row $i$;
22         $D[i] \leftarrow max - min$;
23     end
24     return $D$
25 Function Average_Algorithm($W$):
26     $D \leftarrow \emptyset$;
27     for $i$ in $W$ do
28         $avg \leftarrow$ get the average of row $i$;
29         $min \leftarrow$ get the minimum value of row $i$;
30         $D[i] \leftarrow avg - min$;
31     end
32     return $D$
```

2) Line 4: In this step, we call the selected greedy algorithm function. The greedy algorithm compares all individuals and all the housing providers' weights and returns a value for each homeless individual. These values are used to prioritize one homeless individual in each iteration. The homeless individual is given priority by getting matched to a housing provider that carries the lowest weight (lower weight denotes a better assignment).
3) Line 5- 16: We initiate a new variable *personIndex*. The algorithm finds the maximum value in set D and assign the index position of that value to *personIndex* variable. We then find the index of the housing provider that carries the lowest weight for that individual from $W[personIndex]$ and assign it to *shelterIndex* variable. Next, the assignment variable $a_{personIndex,shelterIndex}$ is set to 1. $H[personIndex]$ is

removed from set H. We decrement $S[shelterIndex]$'s capacity by 1. if $S[shelterIndex]$'s capacity is 0, then $S[shelterIndex]$ is removed from set S. At the end of the while-loop set A is returned.

Algorithm 1 : *Greedy* − *Part* 2 includes four greedy algorithms as functions. A detailed explanation of those functions is as follows:

- Line 1 to 8: For every row i in set W, the *STDEV_Algorithm* calculates the standard deviation of the set $W[i]$ and stores the result in *stdev* variable. Next, the algorithm searches for the smallest weight in the same row and stores the weight in the *min* variable. In the last step, the algorithm subtracts the *min* from *stdev* and stores the result to $D[i]$.
- Line 9 to 16: For every row i in set W, the *Median_Algorithm* searches for the median value of the set $W[i]$ and assigns the value to the *mdn* variable. Next, the algorithm searches for the smallest weight in the same row and stores the weight in the *min* variable. In the last step, the algorithm subtracts the *min* from *mdn* and stores the result to $D[i]$.
- Line 17 to 24: For every row i in set W, the *Minmax_Algorithm* searches for the minimum and the maximum weights in row $W[i]$ and assigns the values to *min* and *max* variables respectively. In the last step, the algorithm subtracts the *min* from the *max* variable and stores the result to $D[i]$.
- Line 25 to 32: For every row i in set W, the *Average_Algorithm* calculates the average value of the set $W[i]$ and stores the result in the *avg* variable. Next, the algorithm searches for the smallest weight in the same row and stores the weight in the *min* variable. In the last step, the algorithm subtracts the *min* from *avg* and stores the result to $D[i]$.

1) WORKING EXAMPLE

In this subsection, each algorithm is executed as a separate program instance, and the step-by-step working example of the algorithms is illustrated and compared using the input data defined in Scenario 1 (Figure 1). Scenario 1 is defined as follows:

Scenario 1: $H = [h_1, h_2, h_3, h_4]$, $S = [s_1, s_2, s_3, s_4]$, $C = [1, 1, 1, 1]$, and $W = [[20, 22, 6, 7], [25, 6, 23, 14], [17, 1, 0, 2], [4, 3, 18, 25]]$.

Here we explain the operation of the algorithms:

- Line 1 to 3: Each algorithm starts by initializing set A and begins the while-loop.
- At the next step, any of the algorithms that are defined in *Algorithm* 1 : *Greedy* − *Part* 2 initiates a for-loop for every row i in set W. Within the loop, the dispersion value for each row is calculated. The values are stored in set D. At the end of the loop, the output variable is returned to *Algorithm* 1 : *Greedy* − *Part* 1. The output array is stored in set D. In the first iteration, D contains the following values:

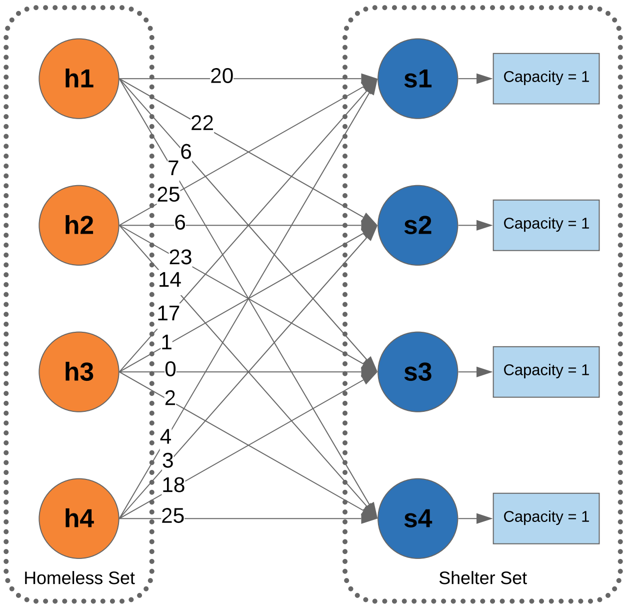


FIGURE 1. Scenario 1.

1) STDEV_Algorithm: $D = [2.4, 2.7, 8.0, 7.7]$
2) Median_Algorithm: $D = [7.5, 12.5, 1.5, 8.0]$
3) Minmax_Algorithm: $D = [16, 19, 17, 22]$
4) Average_Algorithm: $D = [7.75, 11.0, 5.0, 9.5]$

- Line 5 to 16: The index positions in set D are similar to those of set H. For example, the value at index $D[0]$ belongs to the element at index $H[0]$. An individual with the highest corresponding value in set D is given priority and matched with a housing provider promptly. For instance, in the case of *STDEV_Algorithm*, homeless individual h_3 has the highest value in set D. Therefore, h_3 is assigned to the housing provider s_3, which carries the minimum *weight* for that individual. Next, the assignment variable $a_{3,3}$ is set to true. The homeless individual h_3 is removed from the set H. Row w_3 is removed from the *weight* set W. The housing provider s_3's capacity is decremented by 1, and since the capacity of s_3 is 0, s_3 is removed from the housing provider set S.

In the next iterations, the same operations are repeated until either the homeless individuals set H, or the housing providers set S is empty. To avoid redundancy, we refrain from explaining the similar steps in the next iterations; however, the returned values in the next two iterations are as follows:

1) a) STDEV_Algorithm : $D = [1.1, 3.5, null, 9.4]$
   b) Median_Algorithm : $D = [1, null, 2, 14]$
   c) Minmax_Algorithm : $D = [14, 11, 17, null]$
   d) Average_Algorithm : $D = [5.0, null, 6.3, 11.6]$
2) a) STDEV_Algorithm : $D = [2.1, -6.2, null, null]$
   b) Median_Algorithm : $D = [0.5, null, 1.0, null]$
   c) Minmax_Algorithm : $D = [13, 11, null, null]$
   d) Average_Algorithm : $D = [0.5, null, 1.0, null]$

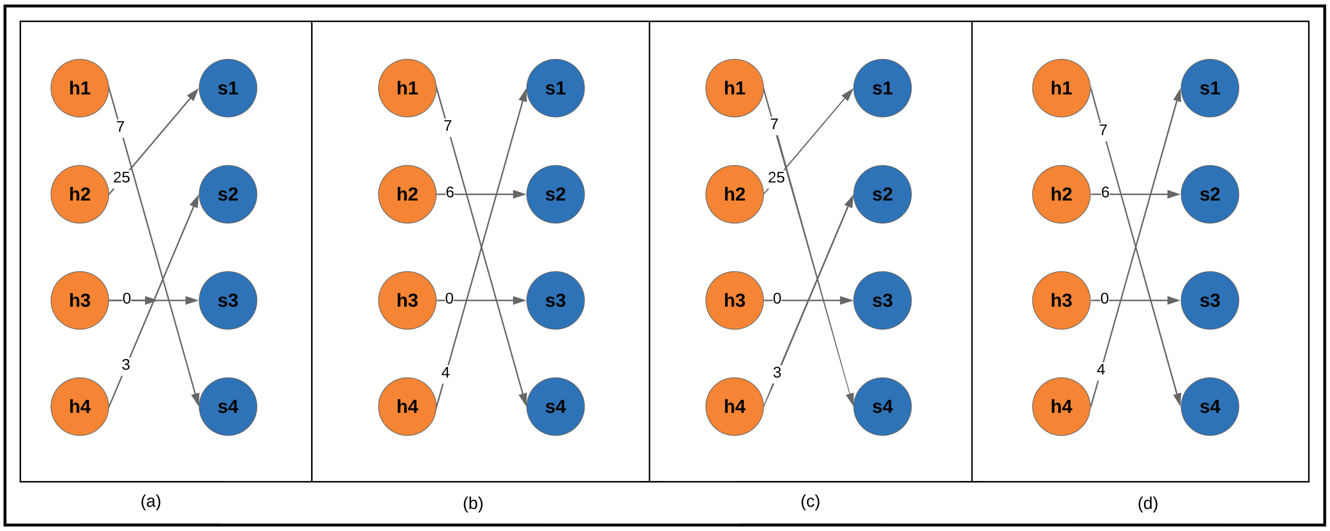


FIGURE 2. **Scenario 1 solution. (a) A solution by STDEV_Algorithm [7, 25, 0, 3]. (b) A solution by Median_Algorithm [7, 6, 0, 4]. (c) A solution by Minmax_Algorithm [7, 25, 0, 3]. (d) A solution by Average_Algorithm [7, 6, 0, 4].**

- The last two homeless individuals are matched with a housing provider one after another. Figure 2 shows the assignment set A that is returned by each algorithm. The assignment set A is as follows:
  1) STDEV_Algorithm : $A = \{a_{3,3}, a_{4,2}, a_{1,4}, a_{2,1}\}$
  2) Median_Algorithm : $A = \{a_{2,2}, a_{4,1}, a_{3,3}, a_{1,4}\}$
  3) Minmax_Algorithm : $A = \{a_{4,2}, a_{3,3}, a_{1,4}, a_{2,1}\}$
  4) Average_Algorithm : $A = \{a_{2,2}, a_{4,1}, a_{3,3}, a_{1,4}\}$

2) WORST CASE

The main feature of the introduced greedy algorithms is the trade-off between the running time and the solution accuracy. Because of that trade-off, greedy algorithms can produce unsatisfactory results. In this section, we give a few examples that illustrate some of the shortcomings of our greedy algorithms. The *Minmax_Algorithm* and the *Median_Algorithm* fail to take every weight into account while making a decision. This can be better explained using the following example: Let us assume that the algorithms are given a single row of weights $w_i = [1, 8, 10, 11, 200]$. The *Minmax_Algorithm* subtracts the minimum weight ($w_{i,1} = 1$) from the maximum weight ($w_{i,5} = 200$) and stores the value in set D. The *Median_Algorithm* subtracts the minimum value ($w_{i,1} = 1$) from the median value ($w_{i,3} = 10$) and similarly stores the results in set D. Both of these algorithms fail to take other shelter weights into account and only measure two values. Similarly, the *STDEV_Algorithm* and the *Average_Algorithm* fail to give a certain weight higher importance. For example, given a single row of weights $w = [1, 2, 3, 100, 200]$. In this case, both algorithms return a high dispersion value regardless of the second and third suitable housing providers (with low weights). These deficiencies can be efficiently improved using local search algorithms. In the next subsection, we explore a Swap-Based local search algorithm to improve the output solution of the greedy algorithms.

B. LOCAL SEARCH METHOD

The local search algorithm takes a feasible solution to the problem, which is returned by the greedy algorithms and repeatedly implement small changes to improve the results. In every iteration, the local search algorithm finds a homeless individual who was given the worst housing provider and swap the individual's housing provider with other homeless people in set H. If, after the swap, the *maximum assigned weight* of the entire solution is minimized, the algorithm updates the solution set and repeats the iteration. The local search algorithm has polynomial running time and achieves a substantial improvement, as shown in the Complexity Analysis subsection. *Algorithm* 2 : *localsearch* is explained as follows:

1) Line 1 to 4: At the first step, we initiate the variable *flag* with initial value *True*. We begin a while-loop that terminates when *flag* is *False*. Within the loop, we set the *flag* to *False* and initiate the variable *MaximumAssignedWeight* with initial value 0.
2) Line 5 to 12: We begin a for-loop that runs for each $a_{i,j}$ in set A. Within the loop, we access set A and find the assignment with the maximum weight and assign that weight to *MaximumAssignedWeight* variable. Similarly, we store the person's index (i) and the shelter's index (j) of that assignment to *VPIndex* and *VPShelterIndex* respectively. (this loop returns the maximum weight in the assignment set A that we are attempting to minimize)
3) Line 13 to 15: In the next step, we initiate a for-loop for every element p in set H. Within the loop, if p is not equal to *VPIndex* (person p is not the same person as the vulnerable person (VP)), then we access A and retrieve the shelter's index (j) that was assigned to p.
4) Line 16 to 17: Next, we access the weight set W and retrieve the weight of the shelter that was assigned to

Algorithm 2 Local Search

```
Input: $W$ : $[[w_{1,1}, w_{1,2}, \ldots, w_{1,n}],$
       $[w_{2,1}, w_{2,2}, \ldots, w_{2,n}], [w_{m,1}, w_{m,2}, \ldots, w_{m,n}]]$;
  $H$ : $[h_1, h_2, \ldots, h_m]$;
  $S$ : $[s_1, s_2, \ldots, s_n]$;
  $A$ ; // assignment solution (output) of
  a greedy algorithm
Output: $A$
1  $flag \leftarrow True$;
2  while $flag$ do
3      $flag \leftarrow False$;
4      $MaximumAssignedWeight \leftarrow 0$;
5      for each $a_{i,j} \in A$ do
           // this loop returns the maximum
              weight in set $A$ which we are
              trying to minimize
6          $w \leftarrow W[i][j]$;   // get the weight of
           this assignment
7          if $MaximumAssignedWeight < w$ then
8              $MaximumAssignedWeight \leftarrow w$;
9              $VPIndex \leftarrow i$;  // person $i$'s index
10             $VPShelterIndex \leftarrow j$;   // person $i$'s
               shelter index
11         end
12     end
13     for each $p \in H$ do
14         if $p \neq VPIndex$ then
15             $pShelterIndex \leftarrow j$; // shelter $j$ that
               was assigned to $p$
16             $swapA \leftarrow W[p][VPShelterIndex]$;
17             $swapB \leftarrow W[VPIndex][pShelterIndex]$;
18             if both swapA and swapB were smaller than
               $MaximumAssignedWeight$ then
19                 $A \leftarrow$ swap the vulnerable person' shelter
                   with $p$'s shelter;
20                 $flag \leftarrow True$;
21             end
22         end
23     end
24 end
```

VP for *p*. Similarly, from W we retrieve the weight of the shelter that was assigned to *p* for *VP*. These two weights are stored in *swapA* and *swapB* respectively.

5) Line 18 - 20: If *swapA* and *swapB* weights are smaller than *MaximumAssignedWeight* that means the highest weight in the assignment solution A was reduced. If so, we will update the assignment solution with the new shelters and set the *flag* to *True*.

C. COMPLEXITY ANALYSIS

Algorithm 1: Greedy Part 1 and Part 2: In this subsection, we provide the complexity analysis of the greedy algorithm and the local search algorithm. The worst case running time of *Algorithm* 1 : *Greedy* − *Part* 1 is $O(m^2 + (m * n))$. The breakdown of this analysis is as follows:

- Line 3: *Greedy* − *Part* 1 runs for every homeless individual in set H. Therefore, in the worst case, the running time of *line* 3, *Part* 1 is $O(m)$.
- Line 4: This line is calling the functions defined in *Greedy* − *Part* 2. We find that every greedy algorithm defined in this research has the worst case running time of $O(m * n)$.
- Line 5: The algorithm searches through set D. Set D contains one value for every element in set H. Therefore, in the worst case, the running time of this procedure is $O(m)$.
- Line 6, 8, 9, 12, and 13: The worst case running time complexity of accessing an array is $O(1)$. Therefore, the running time complexities of these lines are $O(1)$.
- Line 7: The program accesses a predefined row index and searches within that row. The row contains one value for every housing provider in set S. Therefore, the worst case running time of this line is $O(n)$.
- Line 10,11: In the worst case, the running time for a deletion procedure is $O(m)$. Therefore, each line has a running time of $O(m)$.
- Line 15: This line does not run in every iteration. However, in the worst case, this line will be executed in every iteration with the running time of $O(n)$.

Algorithm 2: Local Search: The worst case running time of the local search algorithm is $O(K * m^2)$:

- Line 2: The while-loop requires a complexity of $O(K)$. Where K is the number of iteration of the while-loop. K is defined as $n * (max(w_{i,j}) - min(w_{i,j}))$. In the worst case scenario, the greedy algorithm returns a solution set where the *maximum assigned weight* is equal to the $max(w_{i,j})$ (maximum weight in set W). Since the local search algorithm improves the solution set by at least 1 weight in every iteration (for every person), then in the worst case the while-loop runs for $n*(max(w_{i,j}) - min(w_{i,j}))$ times.
- Line 5 ∼ 13: Line 5 is repeated for every element in set A. Since the assignment solution possibly contains an assignment for every homeless individual then $|A| \leq |H|$. Therefore, Both lines are executed for every element i in set H. The worst case running time for each of these lines is $O(m)$.
- Line 19: The worst case running time for the array insertion is $O(m)$. Therefore worst case running time of this line is $O(2 * m)$ which requires two insertion.
- Other lines of the algorithm have the worst case running time of $O(1)$.

V. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETUP

We perform the experiments on a Windows machine, with an Intel Core i5-8500 CPU and 8.00GB of memory. Algorithms were developed in Python version 3.6.5, and the ILP

TABLE 1. Performance comparison.

Algorithms	n=100	n=300	n=500	n=700	n=1000	n=1300
Optimal Solution	25	22	21	21	20	20
STDEV Algorithm	79	83	68	55	81	57
Median Algorithm	40	52	50	42	38	32
Minmax Algorithm	70	76	43	54	76	58
Average Algorithm	51	35	41	40	38	38
STDEV + local search	34	28	25	24	24	26
Median + local search	34	27	27	26	24	25
Minmax + local search	38	27	26	24	24	26
Average + local search	**33**	**27**	**24**	**24**	**23**	**23**

TABLE 2. CPU computation time comparison (HH:MM:SS:FF).

Algorithms	n=100	n=300	n=500	n=700	n=1000	n=1300
Optimal Solution	0:0:02:62	0:15:22:59	02:26:49:37	07:46:33:22	23:41:52:23	38:03:40:78
STDEV Algorithm	0:0:0:13	0:0:3:44	0:0:15:88	0:0:42:60	0:2:6:51	0:4:30:16
Median Algorithm	0:0:0:036	0:0:1:06	0:0:5:16	0:0:15:04	0:0:46:29	0:1:43:18
Minmax Algorithm	0:0:0:013	0:0:0:27	0:0:1:18	0:0:3:18	0:0:9:28	**0:0:21:06**
Average Algorithm	0:0:0:04	0:0:1:12	0:0:5:82	0:0:14:26	0:0:40:55	0:1:26:79
STDEV + local search	0:0:0:1305	0:0:3:48	0:0:15:99	0:0:42:90	0:2:6:98	0:4:30:87
Median + local search	0:0:0:04	0:0:1:08	0:0:5:18	0:0:15:16	0:0:46:80	0:1:43:98
Minmax + local search	0:0:0:0139	0:0:0:31	0:0:1:26	0:0:3:52	0:0:9:92	**0:0:22:07**
Average + local search	0:0:0:0404	0:0:1:13	0:0:5:90	0:0:14:44	0:0:41:02	0:1:27:32

TABLE 3. Iteration count (local search algorithm).

Algorithms	n=100	n=300	n=500	n=700	n=1000	n=1300
STDEV + local search	27	36	41	45	59	77
Median + local search	9	16	12	18	30	35
Minmax + local search	7	44	40	54	55	51
Average + local search	5	10	21	24	34	42

solver was developed using IBM ILOG CPLEX Optimization Studio [23]. This research is interested in the performance accuracy, run-time, and fairness of the algorithms for large enough population. The algorithms are compared for different population sizes (100, 300, 500, 700, 1000) and randomly generated weights for the problem defined in (6b) to (6f).

B. RESULT COMPARISONS

The optimal solution for each scenario was obtained using the ILP solver. To examine the accuracy of our algorithms, we compare each of the solutions to that of the ILP solver. Furthermore, the algorithms are compared in terms of the assignment weights. An algorithm with the lowest *maximum assigned weight* is considered a better algorithm. Table 1 shows the *maximum assigned weight* (an assignment with the maximum weight) within the solution set which was produced by each algorithm. In terms of the *maximum assigned weight* metric the *Average + local search* algorithm had the best results. Table 1 also shows the performance difference between the algorithms with and without the Local-Search procedure. It can be observed that the Local-Search algorithm significantly improved the performance of the greedy algorithms. As discussed in the Complexity Analysis subsection, in the worst case scenario, the Local-Search algorithm requires at least $K = n * (max(w_{i,j}) - min(w_{i,j}))$ iterations to improved the solution set. However, Table 3 shows the iteration count of each Local-Search algorithm (until termination) where the number of steps was significantly lower than the worst case scenario. Figure 6 illustrates the average accuracy of each algorithm. The *Average + local search* algorithm obtained 92% accuracy followed by *STDEV + local search* (91.6%) and *Median + local search* algorithm (91.2%). Using the run-time comparisons that are presented in Table 2, it can be observed that our algorithms are significantly faster than the solver's program. For example, in the case of the *Minmax + local search* algorithm (1300 homeless individuals), the execution time was improved from 136800 to 21 seconds. To examine the relationship between the number of shelters and algorithm's performance we created several scenarios varying the number of shelters (n = [10, 50, 100, 150, 200, 250, 300, 350, 400]), with a fixed number of homeless individuals (m = 400). For every number of shelters n, we created three scenarios (randomly generated weights and capacities). The capacities were randomly distributed across the shelters. To create a feasible solution the shelter's capacity matched the number of individuals in every scenario. In total, there were 27 scenarios.

To compare the results of the algorithms we collected the *maximum assigned weight* of each solution. Figure 4 shows the results of this comparison. It is noticeable that after a certain number of shelters, the performance of the algorithms did not continue to improve. These observations can help in terms of shelter capacity utilization.

Furthermore, fairness comparison was performed based on Jain's Fairness Index [24]. Jain's Fairness Index provides strong feedback on equality of all assigned weights among homeless individuals, but not their magnitude. It is important to note that the optimal solution produced by the solver is not necessarily a fair assignment since fairness maximization was not the objective of this research. Figure 5 shows the Fairness Index of the ILP solver and other algorithms. The ILP solver obtained a Fairness Index of 99.90%, followed by the *Average + local search* algorithm (99.75%) and the *Median + local search* algorithm (99.49%). Based on all observations, it is evident that the *Average+local search* algorithm performed better than other algorithms.

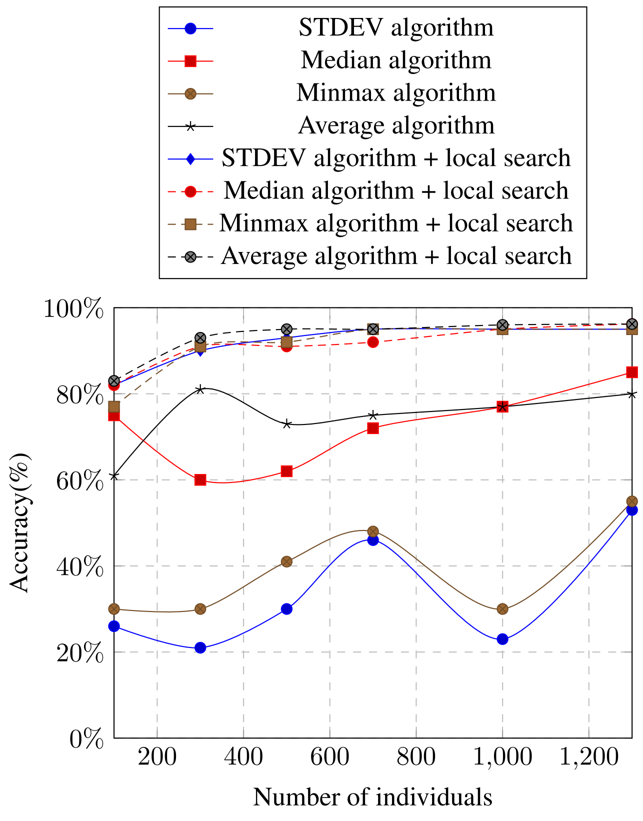


FIGURE 3. Algorithms' accuracy vs the number of individuals (100, 300, 500, 700, 1000, 1300).

Figure 3 presents accuracy comparisons of the proposed algorithms for a different number of homeless individuals. Similar to the previous comparisons the accuracy is measured in relevance to the optimal solution provided by the ILP solver. We can conclude that a larger homeless population did not have a negative effect on the performance of our algorithms. The source code of the experiments can be found on our GitHub repository (https://github.com/pedramvdl31/Smart-City-Response-to-Homelessness.git).

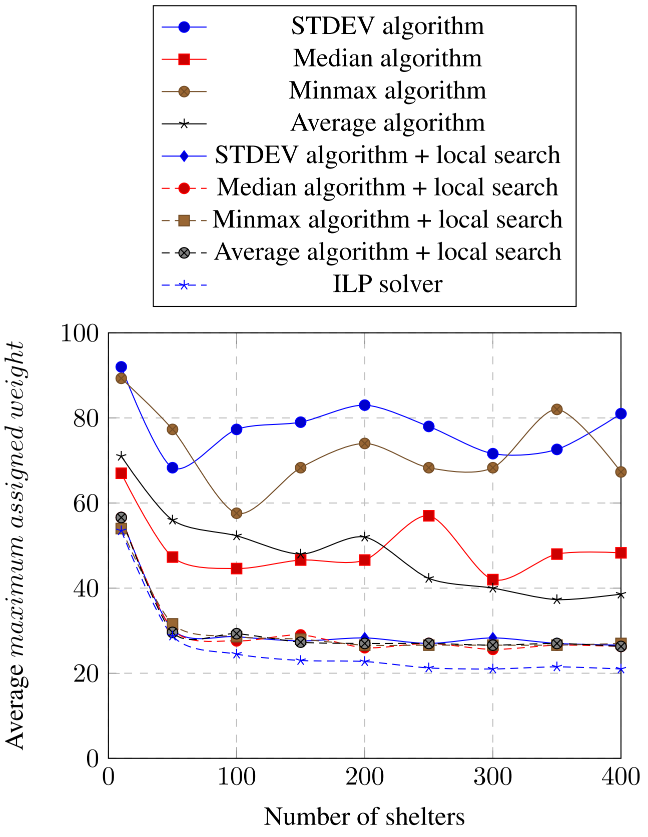


FIGURE 4. Algorithms' performance vs the number of shelters (10, 50, 100, 150, 200, 250, 300, 350, 400).

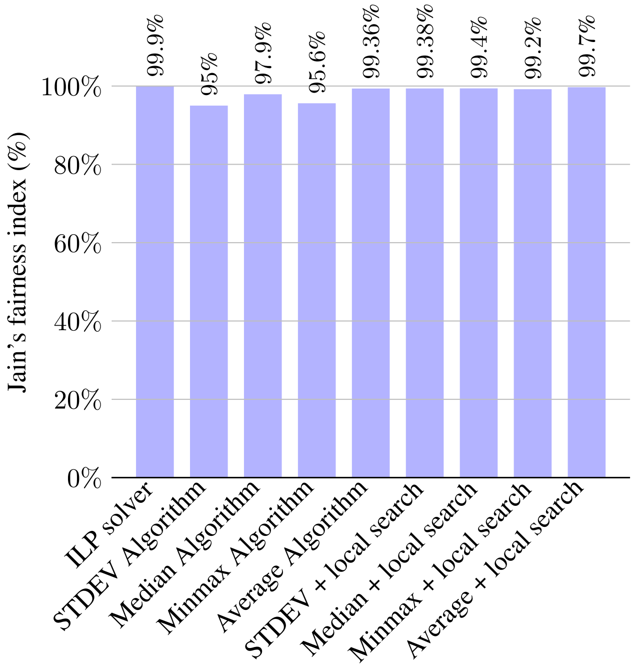


FIGURE 5. Fairness index comparison.

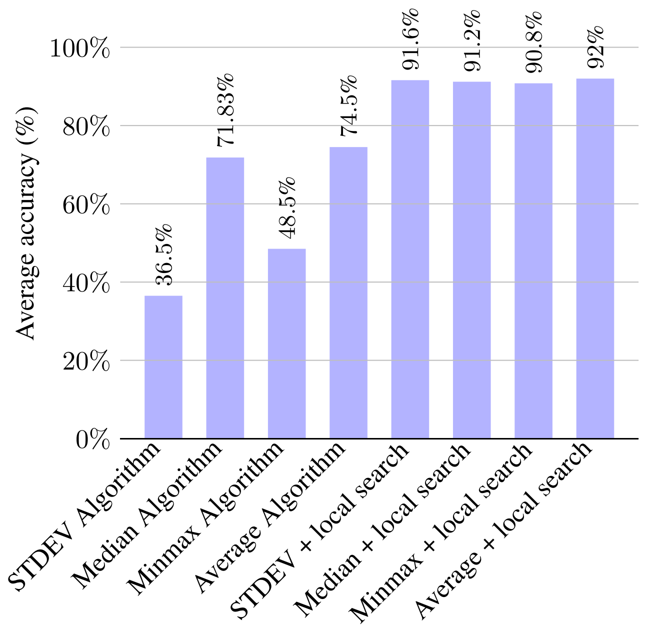


FIGURE 6. Average accuracy comparison.

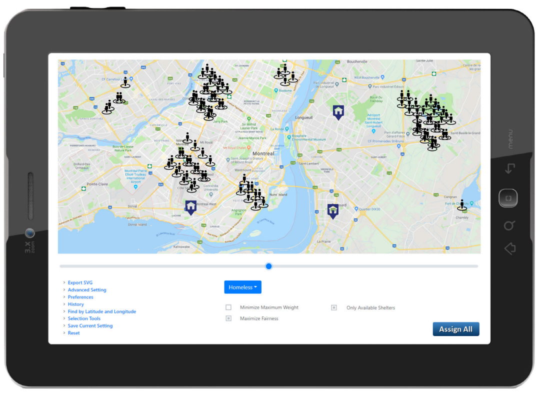


FIGURE 7. Policymaker application.

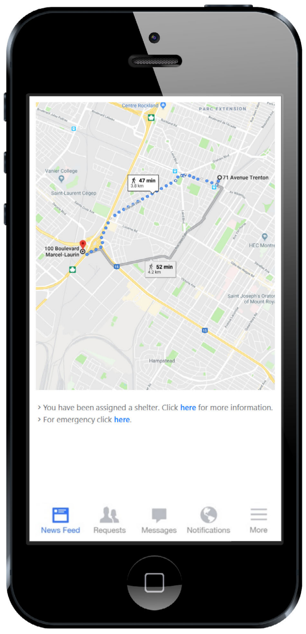
VI. CONCLUSIONS AND FUTURE WORK

The homeless population is highly diverse. Currently, in Canada, housing providers typically provide different types of services to individuals with different needs. However, the diversity of the population, their geographical dispersion across cities, and other circumstances make the homeless-to-housing matching a complex task. Our works aims to lay a foundation for the development of a platform to facilitate accomplishing it more efficiently.

We believe our proposal is realistic. In the United States, the UK, and at least many countries in continental Europe in addition to Canada, emergency shelters and transitional housing providers are usually owned by a variety of non-profits, which can be large or small. For our application to be realizable, two conditions need to be met: (1) each provider needs to update information about remaining beds in an electronic system in real-time; and (2) information about remaining spaces contained in each electronic system needs to be gathered and made available, also in real-time. The first of these conditions seems to us likely to already be met by many providers. It is in the provider's interest that individuals already registered for the night, or for whom a place is being reserved, be recorded in an electronic system so that: (a) the individual at the front desk can keep track of how many spaces remain available, and of what types, at any given moment; and (b) when the time comes to prepare reports for the government or funders, including information about occupancy rates, etc., these can be quickly put together. The second condition is not, to our knowledge, currently met. However, considering existing integrated online platforms, e.g., in commercial applications, it is quite feasible technically. It is only a matter of government and providers deciding to implement a proposal such as ours. We also note that homeless individuals who do not have a smartphone (surprisingly perhaps, a number do), someone at the front desk of a provider could access the system and be able to quickly and reliably refer the individual to an appropriate resource with available space. We note also that the recent growing interest in smart cities [1] invites the development of such a platform. In this research, we introduced several algorithms that produce suitable homeless-to-housing matching. Besides maximizing the Goodness-of-fit, fairness, algorithms accuracy (in comparison with the optimal solution), and the computational time figured among our objectives. Algorithms that were introduced in this research delivered satisfactory results. Our best algorithm ($Average + localsearch$) produces 92% accuracy, 99.75% fairness, and approximately reduces the computation time by 38 hours. An interesting extension of this work would be a bi-objective algorithm that maximizes the Goodness-of-fit while maximizing the Fairness Index. We are also interested to see the Smart Housing Framework in practice. Smart Housing Framework is potentially a complete platform consists of the assignment algorithm program, a policymaker control application, and a specialized application for homeless individuals. Figures 7 and 8 are a representation of what we envision for our future real-time application. Figure 7 shows the application for policymakers that provides all the necessary tools and statistical data in order to assist the policymakers in the decision making process. Figure 8 shows the phone application for homeless individuals who receive notifications upon a new assignment. Additionally, the phone application can assist the individual with the walking direction to the housing provider.

Other extensions of this work and extra features that can be beneficial to policymakers and homeless individuals are as follows:

- The ability to access the third-party API in order to retrieve additional information and a history report about the homeless individuals on the map.
- Providing analytical tools for the policymakers.
- Providing homeless growth predictions based on the available data.
- Performing shelter capacity utilization analysis, which can be used by policymakers to justify opening a new housing site or to close an existing site.



› You have been assigned a shelter. Click here for more information.
› For emergency click here.

FIGURE 8. **Client/homeless individual application.**

- Scheduling application to assist the homeless individual in future planning.
- Swapping, or moving homeless individuals from one housing provider to another to increase the overall Goodness-of-fit.

ACKNOWLEDGMENT

The authors would like to thank A. Garg, A. Fisher, Dr. B. Gajderowicz, and D. Chandrasekaran of the DaTALab (https://www.datalab.science) for proofreading the article and providing inputs.

REFERENCES

[1] J. Aráuz, "Smart Cities and the Dire Need for a Course Correction," in *Proc. IEEE Int. Smart Cities Conf. (ISC2)*. Kansas City, MO, USA, Sep. 2018, pp. 1–6.

[2] M. L. Akinluyi and A. Adedokun, "Urbanization, environment and homelessness in the developing world: The sustainable housing development," *Medit. J. Social Sci.*, vol. 5, no. 2, p. 261, 2014.

[3] R. Y. M. Li and J. Li, "The impact of subprime financial crisis on Canada and United States housing market and economy," in *Proc. ICBMG Conf.*, Hong Kong, vol. 59, 2013, pp. 1–6.

[4] S. Gaetz, T. Gulliver, and T. Richter, "The state of homelessness in Canada 2014: Canadian homelessness research network," Homeless Hub Press, Toronto, ON, Canada, Tech. Rep. 978-1-77221-001-9, Jan. 2014.

[5] J. S. Roncarati, T. P. Baggett, J. J. O'Connell, S. W. Hwang, E. F. Cook, N. Krieger, and G. Sorensen, "Mortality among unsheltered homeless adults in Boston, Massachusetts, 2000-2009," *JAMA Internal Med.*, vol. 178, no. 9, pp. 1242–1248, Sep. 2018, doi: 10.1001/jamainternmed.2018.2924.

[6] E. Latimer and F. Bordeleau, "Dénombrement des personnes en situation d'itinérance au québec le 24 avril 2018," Ville de Montreal et Centre intégré universitaire de santé et de services sociaux du Centre-Sud-de-l'île-de-Montreal, Montreal, QC, Canada, Tech. Rep. 18-846-10W, Mar. 2019.

[7] E. A. Latimer, D. Rabouin, Z. Cao, A. Ly, G. Powell, T. Aubry, J. Distasio, S. W. Hwang, J. M. Somers, V. Stergiopoulos, S. Veldhuizen, E. E. Moodie, A. Lesage, and P. N. Goering, "Costs of services for homeless people with mental illness in 5 Canadian cities: A large prospective follow-up study," *CMAJ Open*, vol. 5, no. 3, pp. E576–E585, Jul. 2017.

[8] J. Kleinberg and E. Tardos, *Algorithm Design*. London, U.K.: Pearson, 2006.

[9] J. T. Marshall and J. Venegas, "The smart cities movement and advancing the international battle to eliminate homelessness-Barcelona as test case," *Revista de Derecho Urbanístico y Medio Ambiente*, to be published.

[10] M. Piat, L. Polvere, M. Kirst, J. Voronka, D. Zabkiewicz, M.-C. Plante, C. Isaak, D. Nolin, G. Nelson, and P. Goering, "Pathways into homelessness: Understanding how both individual and structural factors contribute to and sustain homelessness in Canada," *Urban Stud.*, vol. 52, no. 13, pp. 2366–2382, Oct. 2015.

[11] T. V. Kumar and B. Dahiya, "Smart economy in smart cities," in *Smart Economy Smart Cities*. Singapore: Springer, 2017, pp. 3–76.

[12] S. Marvin, A. Luque-Ayala, and C. McFarlane, *Smart Urbanism: Utopian Vision Or False Dawn?*. Evanston, IL, USA: Routledge, 2015.

[13] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, "Current trends in Smart City initiatives: Some stylised facts," *Cities*, vol. 38, pp. 25–36, Jun. 2014.

[14] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for Internet of everything," *China Commun.*, vol. 16, no. 3, pp. 32–41, Mar. 2019.

[15] A. W. de Jong, J. I. Rubrico, M. Adachi, T. Nakamura, and J. Ota, "Big data in automation: Towards generalized makespan estimation in shop scheduling problems," in *Proc. 13th IEEE Conf. Automat. Sci. Eng. (CASE)*. Xi'an, China, Aug. 2017, pp. 1516–1521.

[16] S. M. H. Hojjati and A. Sahraeyan, "Minimizing makespan subject to budget limitation in hybrid flow shop," in *Proc. Int. Conf. Comput. Ind. Eng.*, Troyes, France, Jul. 2009, pp. 18–22.

[17] P. Singh, M. Dutta, and N. Aggarwal, "Bi-objective HWDO algorithm for optimizing makespan and reliability of workflow scheduling in cloud systems," in *Proc. 14th IEEE India Council Int. Conf. (INDICON)*, Roorkee, India, Dec. 2017, pp. 1–9.

[18] M. Khodier and C. Christodoulou, "Linear array geometry synthesis with minimum sidelobe level and null control using particle swarm optimization," *IEEE Trans. Antennas Propag.*, vol. 53, no. 8, pp. 2674–2679, Aug. 2005.

[19] A. A. Masmoudi and M. Benbrahim, "New heuristics to minimize makespan for two identical parallel machines with one constraint of unavailability on each machine," in *Proc. Int. Conf. Ind. Eng. Syst. Manage. (IESM)*, Seville, Spain, Oct. 2015, pp. 476–480.

[20] J. Huang and L. Wang, "Makespan minimization on single batch-processing machine considering preventive maintenance," in *Proc. 5th Int. Conf. Ind. Eng. Appl. (ICIEA)*, Singapore, Apr. 2018, pp. 294–298.

[21] A. W. De Jong, J. I. U. Rubrico, M. Adachi, T. Nakamura, and J. Ota, "A generalised makespan estimation for shop scheduling problems, using visual data and a convolutional neural network," *Int. J. Comput. Integr. Manuf.*, vol. 32, no. 6, pp. 559–568, Jun. 2019.

[22] D. Sapra, R. Sharma, and A. P. Agarwal, "Comparative study of metaheuristic algorithms using Knapsack Problem," in *Proc. 7th Int. Conf. Cloud Comput., Data Sci. Eng.-Confluence*, Noida, India, Jan. 2017, pp. 134–137.

[23] (2014). *IBM: ILOG CPLEX Optimization Studio 12.7.1*. [Online]. Available: http://ibm.biz/COS1271Documentation

[24] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," in *Eastern Research Laboratory*. Hudson, MA, USA: Digital Equipment Corporation, 1984.

[25] V. S. Inampudi and A. Ganz, "Web based tool for resource allocation in multiple mass casualty incidents," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Minneapolis, MN, USA, Sep. 2009, pp. 1710–1713.

[26] A. Alsubaie, K. Alutaibi, and J. Marti, "Resources allocation in emergency response using an interdependencies simulation environment," in *Proc. IEEE Canada Int. Humanitarian Technol. Conf.*, Ottawa, ON, Canada, May/Jun. 2015, pp. 1–4.

PEDRAM KHAYYATKHOSHNEVIS received the bachelor's degree in computer science from University Sains Malaysia, in 2012, and the M.Sc. degree in computer science from the University of Seoul, South Korea, in 2015. He continued working as a Programmer and a volunteer making crowdfunding application to help the victims of trauma and others who were in need. He worked as a Programmer in three successful startups for the length of three years. He moved to Canada as a Computer Science Lab Technician and a Researcher with the University of Regina. At the end of his contract, he moved to Thunder Bay to pursue his education and research as a thesis-based researcher at Lakehead University.

SALIMUR CHOUDHURY received the Ph.D. degree in computing from Queen's University, Kingston, ON, Canada, in 2012. He is currently an Assistant Professor with the Department of Computer Science, Lakehead University, Thunder Bay, ON, Canada. His research interests include designing algorithms for wireless communication, optimization, cellular automata, approximation algorithms, and so on. He was the Technical Program Chair of the SGIoT 2017 Conference. He is also the Technical Program Chair of SGIoT 2018 and iThings 2018 Conferences. He has been serving as an Editor for the *Parallel Processing Letters* and the *International Journal of Computers and Applications*.

ERIC LATIMER received the Ph.D. degree. He is currently a Research Scientist with the Douglas Research Centre and a Professor with the Department of Psychiatry, McGill University, Montreal, QC, Canada. A health economist by training, he has carried out research on mental health services and services for homeless people, for more than 25 years. He has carried out multiple studies on evidence-based practices for people with severe mental illness and homeless people, including Assertive Community Treatment, supported employment, Housing First, and the strengths model for case management. He was the Lead Researcher of the Montreal site of the CAN$ 110 million At Home/Chez soi pan-Canadian trial of Housing First, and the lead economist for the national study. He has also led the analysis of surveys of homeless individuals in the province of Quebec, Canada.

VIJAY MAGO received the Ph.D. degree in computer science from Panjab University, India, in 2010. In 2011, he joined the Modeling of Complex Social Systems Program, IRMACS Centre of Simon Fraser University. He is currently an Associate Professor with the Department of Computer Science, Lakehead University, Thunder Bay, ON, Canada, where he teaches and conducts research in areas, including big data analytics, machine learning, natural language processing, artificial intelligence, medical decision making, and Bayesian intelligence. He has published extensively (more than 50 peer-reviewed articles) on new methodologies based on soft computing and artificial intelligence techniques to tackle complex systemic problems, such as homelessness, obesity, and crime. He has served on the program committees of many international conferences and workshops. In 2017, he joined the Technical Investment Strategy Advisory Committee Meeting for Compute Ontario. He also serves as an Associate Editor of IEEE ACCESS and *BMC Medical Informatics and Decision Making* and the Co-Editor for the *Journal of Intelligent Systems*.