Automatic Layout Design Solution

Conference Paper · January 2011

DOI: 10.1007/978-3-642-24800-9_20 · Source: DBLP

2 authors:

Fadratul Hafinaz Hassan
Universiti Sains Malaysia
27 PUBLICATIONS **73** CITATIONS
SEE PROFILE

Allan Tucker
Brunel University London
187 PUBLICATIONS **1,800** CITATIONS
SEE PROFILE

Automatic Layout Design Solution

Fadratul Hafinaz Hassan[1,2] and Allan Tucker[1]

[1] School of Information Systems, Computing and Mathematics, Brunel University, UK
[2] School of Computer Science, University Science of Malaysia, Penang, Malaysia
{fadratul.hassan,allan.tucker}@brunel.ac.uk

Abstract. A well designed space must consider not only ease of movement, but also ensure safety in a panic situation, such as emergency evacuation in a stadium or hospital. The movement statistics connected with pedestrian flow can be very helpful in feasible layout design. In this study, we examined four-way pedestrian movement statistics generated from heuristic search techniques to find feasible classroom layout solutions. The aim of the experiment is to compare how fast and effective the algorithms can generate automatic solutions to the layout. Experiments from our preliminary study have shown that promising results for simulated annealing and genetic algorithm operator algorithms in pedestrian simulation modelling. Our experimental results are compared with current classroom layouts. We find a feasible layout with a staggering shaped and wider lane, objects shift aside to the walls creating bigger lanes in the middle of layout, or escape routes are created surrounding the clustered objects.

Keywords: Genetic Algorithm, Simulated Annealing, Cellular Automata, Pedestrian statistics, Classroom layout.

1 Introduction

Pedestrian simulations on egress plans provide in-depth analysis on pedestrian movement during evacuation processes but do not offer any consideration of the distribution of objects or obstacles inside the floor layout itself. The smooth flow of pedestrians is formed by a feasible allocation of obstacles in a layout. Thus, in architectural planning, it is important to carefully consider the position of obstacles in a layout as well as the location of exits. However, many pedestrian simulation studies on evacuation plans are often focused on pedestrian behaviour and do not assess the interaction with obstacles inside the floor plan. If we could analyse the relationship of different pedestrians with the obstacles, then it would allow us to configure a better distribution of obstacles in a layout. For example, pedestrian statistics generated from the simulations could be investigated further and show us where the collision points are in a full grid layout. That way, we can identify 'bad' regions and predict feasible layouts. If we could also study different types of pedestrian flow in a microscopic level of the simulation, then it will give us greater control on different sets of pedestrians in and out of the layout. For example, an evacuation process in a hospital, where the allocation of some critical resources, namely medical and non-medical

J. Gama, E. Bradley, and J. Hollmén (Eds.): IDA 2011, LNCS 7014, pp. 198–209, 2011.

staff, to the various units of a hospital at the time of emergency. This way, we have a choice to first evacuate sets of patients as fast as possible and save more lives.

In this paper, we implement simulated annealing (SA) and hill climbing (HC) with updates that involve simple genetic algorithm (GA) operators in order to find solutions to a classroom layout. Previously, we have used these algorithms to simulate a 10-by-10 grid size with two types of pedestrian moving inside the space; one moving to the left and one moving to the right. In the previous experiments, left and right pedestrians kept moving in the same direction until the end of the simulation and without any distinctive exit on the layout [2], [3]. In this work, we improved the current layout by adding static objects such as walls, distributed on every side of the room with two openings on each side, and non-static objects randomly distributed. The size of the room was also increased from 10-by-10 to 20-by-20 grid size. We added two more types of pedestrian moving inside the space: one moving up-to-down and another moving down-to-up based upon the cellular automata model of Yue et al. [12]. The existing fitness function was updated to reflect movement from all four types of pedestrian. In order to create a realistic simulation, we enhanced the pedestrian behaviour by allowing each type of pedestrian to randomly change their direction during the simulation. The aim of the experiments was to compare and understand how fast and effective the algorithms can generate automatic solutions to the spatial layout problem by using statistics generated from cellular automata pedestrian simulations.

This paper is organised as follows. Section 2.1 introduces Hill Climbing (HC), Simulated Annealing (SA) and extended SA using Genetic Algorithm-style operator (SA-GAO). Sections 2.2 and 2.3 describe move operator and fitness function. Section 3 presents experimental results on the classroom layout. Finally we present our conclusions in section 4.

2 Methodology

The experiments involved applying SA, HC and SA-GAO to solve the spatial layout problem. It was not feasible to have a full GA implementation due to the very complex fitness function involving several pedestrian simulations.

2.1 Hill Climbing, Simulated Annealing and SA Genetic Algorithm Operators

HC is a comparatively simple local search algorithm that works to improve a single candidate solution, starting from a randomly selected point. From that position, the neighbouring search space is evaluated. If a fitter candidate solution is found, the search moves to that point. If no better solution is found in the vicinity, the algorithm terminates. The main disadvantage of using HC is that it often gets stuck in local maxima during the search [8].

SA is an extension to HC, reducing the chance of converging at local optima by allowing moves to inferior solutions under the control of a temperature function. This solution is followed if a better successive can be found. Otherwise it accepts a worse state with a certain probability that decreases with temperature. This is less extreme than taking randomised HC each time but still has the ability of escaping from the

possible trap of local maximum/minimum or plateau [8]. The pseudocode for our implementation of this approach is listed below.

```
Input: Number of iterations, iteration, and a random
starting layout, startrep, starting temperature,
temperature
oldrep = startrep;
Apply 10 pedestrian simulations to generate statistics,
stats
Fit = fitness(stats)
bestfit = fit
for        loop = 1:iteration
             rep = oldrep;
             Apply move operator to rep
             Apply 10 pedestrian simulations to stats
             newfit = fitness(stats);
             dscore = newfit-fit
             if ((bestfit < newfit) OR
             (rand(0,1) <e^(dscore/temperature)))
                 bestfit = newfit
             oldrep = rep;
             else
                 rep = oldrep;
             end if
           temperature = temperature*0.9
end for
```

In order to set this as a HC, the starting temperature is set to zero. Note that the fitness here uses the statistics generated from 10 repeated runs of the CA pedestrian simulation. This is done to ensure that one simulation does not result in a 'lucky' fitness score for one layout based upon the starting positions of the pedestrians.

Then we extended our work by using GA-style operators [3]. A full GA implementation was not feasible due to the very complex fitness function involving several pedestrian simulations. The initial 'parents' were selected from the best solutions generated (selections were made based on more consistent fitness values with a fitness higher than the original fitness which was 43.434 or above) from a number of SA experiments. We experimented with different styles of combination for two 'parents' that more or less acted like a uniform crossover.

2.2 Move Operator

The move operator is an enhancement from the previous study[2] and takes into account the size of the simulation grid, randomly moving one object a fraction of this distance (determined by the parameter *changedegree*). The result of the move is then

checked to see if the new coordinates are within the bounds of the grid and do not result in the object overlapping with others. In this update, we also checked the result did not overlap with the static object (wall). The operator is defined below, where *unidrnd(min,max)* is a uniform discrete random number generator with the limits of *min* and *max*. The pseudocode for move operator is listed below.

```
Input: Size of simulation grid, W, size of objects,
sizobj, current x-coordinate, oldx, current y-
coordinate, oldy
Set the degree of change to make based upon fraction of
the grid size:
Changedegree = W/2;
Choose a random object in the grid, i
[oldx, oldy] = current x and y coordinates of object i
xchange = unidrnd(-changedegree/2, changedegree/2)
ychange = unidrnd(-changedegree/2, changedegree/2)
if ((oldy + ychange) and (oldx + xchange) is within
grid boundary AND new object position does not overlap
another object taking into account sizobj)
          newx = oldx + xchange;
          newy = oldy + ychange;
end if
Move object i to position [newx, newy]
Output: newx newy
```

2.3 Fitness Function

The fitness function is calculated based on the statistics that are generated using the pedestrian simulation. The statistics take the form of a 3x3 matrix, *leftstats*, representing the sum of decisions for left-moving pedestrians and a similar decision matrix for right-moving pedestrians; *rightstats*, up-moving pedestrians; *upstats*, and down-moving pedestrians; *downstats*. Therefore, the middle cell in each grid represents how many times the pedestrians decided to stay in the same cell as last time. As we wish to encourage free flow we wish to increase the fitness for layouts that result in many cases of left moving pedestrians moving left, right-moving pedestrians moving right, up-moving pedestrians moving up and down-moving pedestrians moving down whilst penalising the fitness of any decisions where the left-moving pedestrians move right, up-moving pedestrians move down and vice-versa. For example, consider the four *stats* matrices for left, right, up and down pedestrians. It is clear that the *leftstats* reflect a 'good' result as the pedestrians have generally moved in the desired direction more often whereas for *rightstats, upstats* and *downstats* this is not the case.

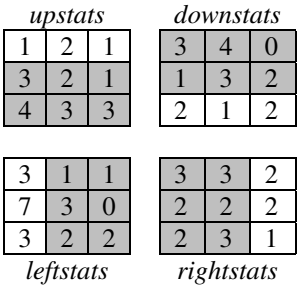


Fig. 1. Up, Down, Left and Right statistics – describe as 3x3 matrix

In general, we wish to maximise the first column in *leftstats,* the third column in *rightstats,* the first row in *upstats* and the third row in *downstats* whilst minimising the other statistics (shaded in the example, Figure 1). Therefore, we use the following fitness function:

$$\text{rightfitness} = \text{sum}(\text{rightstats}(3,1:3)) - \text{sum}(\text{rightstats}(1,1:3)) . \quad (1)$$

$$\text{leftfitness} = \text{sum}(\text{leftstats}(1,1:3)) - \text{sum}(\text{leftstats}(3,1:3)) . \quad (2)$$

$$\text{upfitness} = \text{sum}(\text{upstats}(1:3,1)) - \text{sum}(\text{upstats}(1:3,3)) . \quad (3)$$

$$\text{downfitness} = \text{sum}(\text{downstats}(1:3,3)) - \text{sum}(\text{downstats}(1:3,1)) . \quad (4)$$

$$\text{fitness} = \text{rightfitness} + \text{leftfitness} + \text{upfitness} + \text{downfitness} . \quad (5)$$

Higher fitnesses should reflect simulations whereby pedestrians have moved in the direction that they wish more often.

3 Results

The experiments involved running HC, SA and SA-GAO on the problem of trying to arrange 15 pre-defined objects in a 20x20 grid with 'left' pedestrians, 'right' pedestrians, 'up' pedestrians and 'down' pedestrians. Each type of pedestrian randomly changes his/her direction during simulation. Each algorithm was run 10 times and the learning curves were inspected. The final fitnesses and quality of the layouts were then investigated. Finally, some inspection of sample simulations on the final layouts was carried out in order to find interesting characteristics.

3.1 Summary Statistics

Table 1 shows the minimum, maximum, mean values and standard deviation for the final fitness of each algorithm over 10 experiments, where SA0.5 represents SA with initial temperature of 0.5, SA0.7 represents a temperature of 0.7 and SA0.9 represents a temperature of 0.9. The statistical values of SA0.5 show the robustness of the solutions with the standard deviation of 3.687. The standard deviation is relatively low, which indicates that SA with initial temperature of 0.5 is among the most consistent approach in finding a good solution. However, SA0.9 has the maximum

value of final fitness (the mean is also the highest). It indicates that at the highest temperature, SA can sometimes escape some of the local optima. It even achieves higher fitness when compared to the original layout (43.434). Note that also, SA0.9 generates two fitnesses which are higher than the original value (45.478 and 44.256). None of the fitness values achieves better values than the original layout fitness from for HC, SA0.5 and SA0.7.

Table 1. Summary statistics of final fitness

Method	Min.	Max.	Mean	Std. Dev.
HC	29.170	42.800	36.334	4.165
SA0.5	31.100	42.506	35.685	**3.687**
SA0.7	23.902	40.154	29.764	5.495
SA0.9	27.380	**45.478**	**36.980**	**5.743**

We then expanded our work to SA-GAO. Based on the result above, 'parents' from SA0.9 were selected. We ran 100 further simulations on SA0.9 to find more parents which have better fitness than the original layout fitness. Eight 'parents' with fitness higher than the original fitness (43.434) were selected.

We next tried to experiment with 100 random combinations of mappings. Around 40 children have fitness better than their 'parents'. The highest fitness of the children was 56.876, as shown in the table below, generated from 'child2' with mapping '112121121222111'. Note that the same mapping also generates a better fitness for 'child1' with fitness value of 49.316. The second highest fitness is from 'child1' with

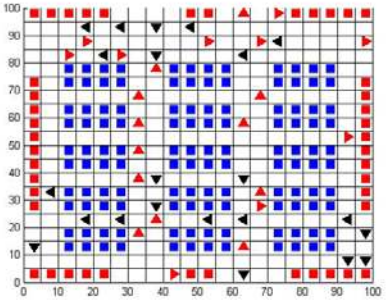
Table 2. Children's mapping with highest fitness values from parents

Child1	Fitness	Child2	Fitness
2 2 2 2 1 1 2 1 1 1 1 2 1 1 1	47.466	2 1 1 1 2 2 2 1 1 1 2 2 1 1 1	53.360
2 1 1 2 1 2 1 1 1 1 2 1 2 1 2	49.330	**2 2 2 2 1 1 2 1 1 1 1 2 1 1 1**	46.970
1 2 2 1 2 2 1 2 2 1 2 2 2 2 2	52.972	2 1 2 2 1 2 1 1 2 1 1 1 1 2 1	48.104
1 2 1 1 1 2 1 2 1 1 1 2 2 1 2	45.756	1 1 2 2 1 2 2 1 2 1 2 1 1 1 2	51.244
1 2 2 1 2 1 1 2 2 1 1 2 2 2 1	46.770	2 1 1 2 2 2 2 2 2 2 2 1 1 2 1	48.536
2 1 1 1 1 2 2 1 2 2 1 1 1 2 2	46.266	1 2 2 1 1 1 1 2 2 1 1 1 1 2 2	46.710
1 2 2 2 2 1 2 2 2 1 2 2 2 2 1	49.276	1 1 1 2 2 1 1 1 2 2 2 1 2 1 2	48.540
1 2 2 1 1 1 2 2 1 1 2 1 1 2 1	52.234	**1 1 2 1 1 2 1 2 1 2 2 1 2 1 1**	48.838
1 1 2 1 1 2 1 2 1 2 2 1 2 1 1	48.590	2 2 2 2 2 2 1 1 2 2 1 1 2 1 1	50.092
2 2 1 1 1 2 2 2 2 2 1 2 1 2 1	45.570	**1 1 2 1 2 1 1 2 1 2 2 2 1 1 1**	**56.876**
1 1 2 1 2 1 1 2 1 2 2 2 1 1 1	49.316	**2 2 1 1 2 2 2 1 1 2 2 2 1 1 2**	46.658
2 2 1 1 2 2 2 1 1 2 2 2 1 1 2	**53.750**	1 1 2 2 2 1 2 2 1 1 2 1 2 1 1	46.826
2 2 1 1 2 1 1 2 1 2 2 1 1 2 1	47.160	**2 1 2 2 1 1 1 2 1 1 1 2 1 2 2**	45.652
2 1 1 2 1 1 1 1 1 2 2 2 2 1 1	48.334	1 2 2 1 1 1 2 2 1 2 2 2 2 1 2	51.728
2 1 2 2 1 1 1 2 2 1 2 2 2 2 1	45.934	2 1 2 2 1 2 2 2 1 2 1 1 1 2 1	50.676
2 1 2 2 1 1 1 2 1 1 1 2 1 2 2	47.692	1 2 2 2 1 2 1 2 2 1 2 1 2 1 1	46.654
2 2 1 1 1 1 1 1 2 2 1 1 2 2 1	45.810	1 2 2 1 1 2 2 2 1 2 2 1 2 2 1	47.006
1 2 2 1 2 1 1 1 1 2 2 2 2 2 1	46.460	**1 2 2 2 2 2 1 1 1 2 1 2 1 2 1**	45.744
1 2 2 2 2 2 1 1 1 2 1 2 1 2 1	46.800		
2 2 2 1 2 1 2 1 1 1 2 1 2 2 1	49.642		

mapping '221122211222112' and its fitness value was 53.75. Note that the same mapping generates a good fitness for 'child2'. Another four mapping breeds both 'child1' and 'child2' with fitness better than their 'parents'. This result shows that it may be necessary to run quite a large number of recombinations of parents to ensure improved fitness. It may also imply that further mutations are required to fine-tune these new solutions. The fitnesses of the 'children' scored better compared with the 'parents' highest fitness value. This implies that recombining the best of the SA solutions can indeed improve the overall layout without having to implement a full GA which would not be feasible for such an expensive fitness function.

3.2 Exploring the Final Layouts with Their Final Pedestrian Statistics

We now explore some of the characteristics of the final layouts discovered by the algorithms. The red square symbols are the permanent walls, the blue squares in Figure 2-4 represent the final positions of random objects; 500 iterations using SA, HC and SA-GAO algorithm. The black and red arrows represented left, right, up and down pedestrians moving in each ways.



upstats

1973	6545	2335
760	102	725
212	225	123

downstats

131	109	112
578	52	592
1916	6111	2199

1484	571	321
5955	386	908
1490	576	309

leftstats

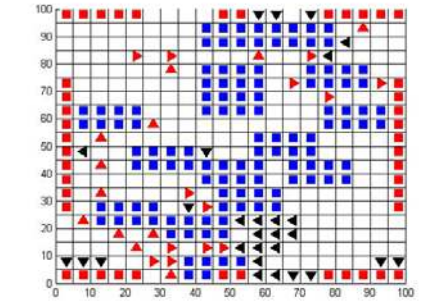
148	708	1593
38	53	**7353**
153	626	1528

rightstats

Fig. 2. Original classroom layout with up, down, left and right statistics

The layout from Figure 2 is based on classical distribution of seats in classrooms [6]. This configuration produced clogging phenomena near the exit [7]. It shows in the layout above where all pedestrians moved and clogged near the exit on the upper right of the room. Exits on the top and bottom walls are not separated far enough; an average exit throughput becomes even significantly less due to a collective slow-down that emerges among pedestrians crossing each other's paths. This disruptive interference effect led the 'up' and 'down' pedestrian to share the same lane [11].

Exits on the left and right room are separated by long walls, thus allow 'left' and 'right' pedestrians to choose different lanes. Notice the 'right' pedestrians only move freely on the upper side of the layout, thus their final statistics, *rightstats* (7353) is the highest among all.



upstats

1790	5071	2002
1103	10	1079
457	69	119

downstats

474	248	295
757	35	741
1904	5767	1979

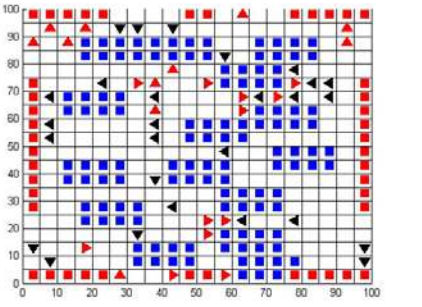
1359	1557	538
4606	**216**	854
1254	1364	952

leftstats

523	1751	2229
112	13	3548
613	1925	1286

rightstats

(a)



upstats

1568	4756	1724
1326	90	1233
504	306	493

downstats

98	416	311
584	31	658
1836	5389	1777

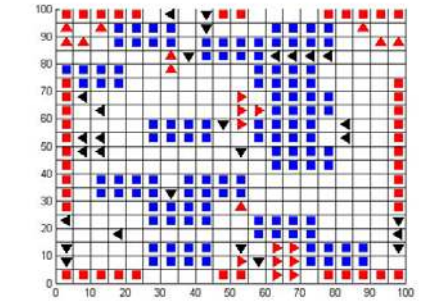
1622	1024	495
5205	**221**	1032
1978	1217	506

leftstats

367	1327	1889
227	**106**	4390
585	1363	1246

rightstats

(b)



upstats

1562	5716	1724
941	136	987
352	531	251

downstats

736	613	671
1294	125	1473
1225	4862	1401

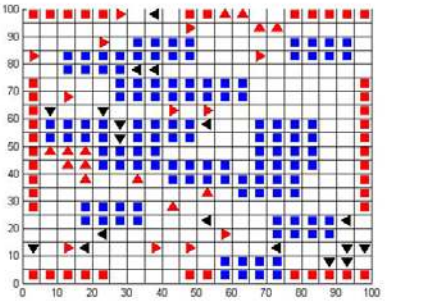
1608	1013	341
5284	109	860
1536	938	210

leftstats

700	1481	1191
257	43	**3773**
703	1577	1375

rightstats

(c)



upstats

2514	5789	1710
589	16	451
106	471	154

downstats

100	22	127
748	21	673
2212	5298	1999

1363	1764	516
3463	593	881
1400	1640	680

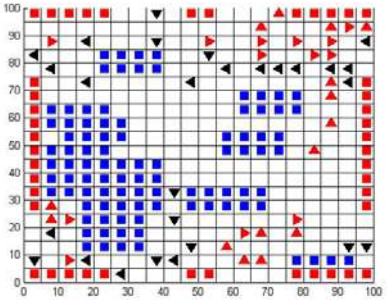
leftstats

948	2124	1434
371	40	2725
1034	1928	1296

rightstats

(d)

Fig. 3. HC and SA final layouts with the lowest final fitness and their final pedestrians statistic, (a) HC (fitness=29.170), (b) SA0.5 (fitness=31.100), (c) SA0.7 (fitness=23.902), (d) SA0.9 (fitness=27.38)



upstats

2294	6688	2168
565	117	490
341	262	575

downstats

258	104	382
817	26	704
2097	5980	1832

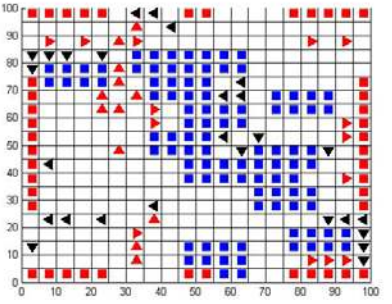
1564	450	27
5677	204	614
2069	516	179

leftstats

297	1172	1947
24	**2**	6202
89	762	2105

rightstats

(a)



upstats

1881	6832	2132
784	21	755
413	244	438

downstats

175	204	139
742	95	933
2676	6234	2002

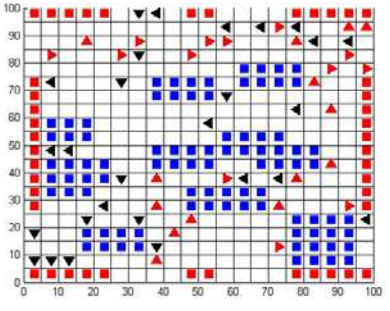
2008	536	165
5940	343	417
1918	705	568

leftstats

184	715	2102
22	9	5461
322	852	2533

rightstats

(b)



upstats

1975	7082	2471
445	28	477
145	367	210

downstats

149	84	166
896	41	914
2063	6070	2217

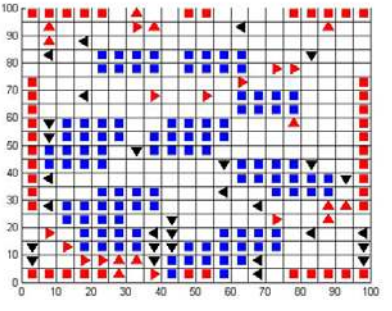
1970	807	289
7170	101	411
2007	606	39

leftstats

230	924	1941
15	**4**	5528
155	1028	2075

rightstats

(c)



upstats

1900	6704	2043
601	32	516
129	90	185

downstats

150	169	285
621	76	765
2103	6074	1957

2556	958	208
4904	62	644
1932	892	644

leftstats

263	1722	1897
55	19	5444
407	1894	1699

rightstats

(d)

Fig. 4. SA-GAO final layouts and their final pedestrians statistic, (a) the highest fitness of 'chi1d1' with fitness of 53.750 and mapping of '221122211222112', (b) 'child2' with same mapping of (a) and fitness of 46.658, (c) 'child1' with the same mapping of (d) and fitness of 49.316, (d) 'child2' with fitness of 56.876 and mapping of '112121121222111' represents the highest fitness of all children

From Figure 3 (a)-(d), we can see clearly that the bad layouts are generated from the series of lowest fitness. Figure 3(a) is the 'bad' layout with the lowest fitness from 10 run of HC. Notice that the objects scattered around in the layout. The objects also create lots of enclosed/ cul-de-sac area which caused the pedestrians being trapped inside. One of eight exits was totally blocked by the objects. There is no clear escape route created. The left and right pedestrians bump to each other because of the narrow path which they have to share. Notice a jammed group of 'left' and 'right' pedestrians in the lower layout where it is a high density location. 'Left' and 'right' have the lowest final statistics and the largest middle value for *leftstats* is 216, In the second layout taken from SA0.5, Figure 3(b) there is jam pattern especially among 'left' and 'right' pedestrians. Notice the bigger middle values for both of *leftstats* and *rightstats* (221 and 106). The third layout, Figure 3(c) shows more than one cul-de-sac that encircled and trapped the pedestrian. This is no surprise since it has the lowest fitness compared to the other three layouts. Notice that 'right' pedestrian stuck in both cul-de-sac resulted their *rightstats* is the lowest compare to the others with only 3773. Around three cul-de-sacs were created in the final layout, Figure 3(d), where the biggest cul-de-sac is in the middle of the layout. One exit on the bottom right of the layout is completely blocked. Clogging among pedestrians can be seen clearly at the high densities region.

The higher the fitness, the better the layout produced as in Figure 4 (a)-(d). All four layouts were generated from SA-GAO simulations. Most of the objects are shifted aside to the walls creating bigger major lanes in the upper side of every layout. The first layout, Figure 4(a) shows the objects move creating a group in one side, giving free spaces near the exits. Therefore, all the pedestrians can move easily without being blocked is shown by the 'right' pedestrians where their *rightstats* middle value reflects it by having the smallest value of 2. Some of the layouts created escape routes that surround the clustered objects as in Figure 4(b). It appears that the 'up' pedestrian leave the virtual trace (chemotaxis) for the behind pedestrian to follow. Figure 4(c) has a bigger lane in the upper layout, allowing pedestrians to move left and right freely. Notice the 'left' and 'right' pedestrians used the straight and bigger path in the upper side of the room that brings out the highest *leftstats* value at 7170 and a small middle value for *rightstats* at 4 (means less pedestrian getting stuck). The fourth layout Figure 4(d), with the highest fitness of all, creates more paths (some zig-zag shape) when compare to other layout. It seems that all the pedestrians move freely and have alternatives to choose any path they prefer. The final statistics for all pedestrians spread averagely even.

4 Conclusions

In this paper we proposed pedestrian flow simulations combined with heuristic searches to assist in the automatic design of classroom layout. Using pedestrian simulations, the activity of crowds can be used to study the consequences of different spatial layouts. Based on the results that have been observed in this paper, we have demonstrated that simple heuristic searches appear to deal with the NP-hard spatial layout design problem to some degree, at least on the very much simplified problem addressed here. The solution is further improved when we paired 'parents' and apply

a GA style operator using our method SA-GAO. Whilst it is not guaranteed that the optimal solution will be found, this does not mean that useful and unexpected designs cannot be discovered using these types of approaches.

Exploring the final pedestrian statistics we found a few characteristics for a feasible layout design. We found that a feasible layout with higher fitness value has better characteristics as zigzag-shaped path, bigger lane and more alternative paths. A zigzag-shaped path can reduces the pressure in panicking crowds and ease the pedestrian movement. A wider lane and several different escape paths help avoid a long waiting lane and clogging effect. It shows in the final statistic where a smooth flow pedestrians has a small middle value of statistics means less waiting or stucking phenomena. Also, we found that where the exits are not separated widely enough, a disruptive interference effect can happen. This scenario results in pedestrians sharing the same path and they have lower statistics when compared to where pedestrians do not share the same path. Indeed, the real positive outcome of the experiments here is that we found certain characteristics that may not have been immediately expected. Future work will be directed towards the gathering of real world data sets and complex systems in order to validate the model equations. The fitness function that will be based on time-to-escape is also a distinct possibility for future study.

References

1. Haklay, M., O'Sullivan, D., Thurstain-Goodwin, M., Schelhorn, T.: "So go downtown": simulating pedestrian movement in town centres. Environment and Planning B 28(3), 343–360 (2001)
2. Hassan, F.H., Tucker, A.: Using Cellular Automata Pedestrian Flow Statistics with Heuristic Search to Automatically Design Spatial Layout. In: 2010 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), p. 32. IEEE, Los Alamitos (2010)
3. Hassan, F.H., Tucker, A.: Using Uniform Crossover to Refine Simulated Annealing Solutions for Automatic Design of Spatial Layouts. In: International Joint Conference on Computational Intelligence (IJCCI), p. 373 (2010)
4. Helbing, D.: A fluid dynamic model for the movement of pedestrians. Arxiv preprint cond-mat/9805213 (1998)
5. Helbing, D., Buzna, L., Johansson, A., Werner, T.: Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transportation Science 39(1), 1 (2005)
6. Helbing, D., Isobe, M., Nagatani, T., Takimoto, K.: Lattice gas simulation of experimentally studied evacuation dynamics. Physical Review E 67(6), 67101 (2003)
7. Kirchner, A., Schadschneider, A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. Physica A: Statistical Mechanics and its Applications 312(1-2), 260–276 (2002)
8. Michalewicz, Z., Fogel, D.B.: How to solve it: modern heuristics. Springer, Berlin (2000)
9. Nishinari, K., Kirchner, A., Namazi, A., Schadschneider, A.: Extended floor field CA model for evacuation dynamics. Arxiv preprint cond-mat/0306262 (2003)
10. Pan, X., Han, C.S., Dauber, K., Law, K.H.: Human and social behavior in computational modeling and analysis of egress. Automation in Construction 15(4), 448–461 (2006)

11. Perez, G.J., Tapang, G., Lim, M., Saloma, C.: Streaming, disruptive interference and power-law behavior in the exit dynamics of confined pedestrians. Physica A: Statistical Mechanics and its Applications 312(3-4), 609–618 (2002)
12. Yue, H., Hao, H., Chen, X., Shao, C.: Simulation of pedestrian flow on square lattice based on cellular automata model. Physica A: Statistical Mechanics and its Applications 384(2), 567–588 (2007)
13. Yue, H., Guan, H., Zhang, J., Shao, C.: Study on bi-direction pedestrian flow using cellular automata simulation. Physica A: Statistical Mechanics and its Applications 389(3), 527–539 (2010)