AN AGENT BASED MODELLING APPROACH FOR THE OFFICE SPACE ALLOCATION PROBLEM

Alexandra Dediu[(a)], Dario Landa Silva[(b)], Peer-Olaf Siebers[(c)]

[(a),(b)]ASAP Research Group, School of Computer Science, University of Nottingham, UK
[(c)]IMA Research Group, School of Computer Science, University of Nottingham, UK

[(a)]alexandra.dediu@nottingham.ac.uk, [(b)]dario.landasilva@nottingham.ac.uk
[(c)]peer-olaf.siebers@nottingham.ac.uk

ABSTRACT
This paper describes an agent based simulation model to create solutions for the office space allocation (OSA) problem. OSA is a combinatorial optimization problem concerned with the allocation of available office space to a set of entities such as people. The objective function in the OSA problem involves the minimization of space misuse and the minimization of soft constraints violations. Several exact and heuristic algorithms have been proposed to tackle this problem. This paper proposes a rather different approach by decomposing the problem into smaller goals, which are delegated to individual agents each representing an entity in the problem. Agents have an internal decision making process which guides them throughout their search process for a better allocation (room). That is, agents seek to satisfy their individual requirements in terms of room space and constraints. Computational experiments show that the agent based model exhibits competitive performance in terms of solution quality and diversity when compared to neighborhood search heuristics.

Keywords: agent based modelling, office space allocation, problem decomposition, distributed optimization

1. INTRODUCTION
Office Space Allocation is a combinatorial optimization problem concerned with the planning and distribution of the available physical space (e.g. rooms) to a set of entities such as people. When planning the distribution of office space in buildings, administrators or managers have to consider the practical issues involved in the allocation process, such as the spatial needs, proximity relations, facilities infrastructure, as well as occupancy costs and effectiveness of the working environment. The aim of OSA is to find an allocation (entities assigned to rooms) such that the misuse of space is minimized, and the satisfaction of constraints is maximized. The OSA problem is faced by many organizations, both academic institutions and businesses, where entities need to be distributed over an often limited amount of room space. Examples of such scenarios where OSA has been investigated as an optimization problem are the NASA Langley Research Center (Kincaid et al. 2007), the European Space Research and Technology Center (Lopes and Girimonte 2010), and dormitories in an academic institution (Trung, Tuan and Anh 2009). So far, this problem has been tackled using various heuristic algorithms, such as asynchronous cooperative local search (Landa-Silva and Burke 2007), nature-inspired algorithms such as harmony search (Awadallah et al. 2012), evolutionary algorithms (Adewumi and Ali 2010; Ülker and Landa-Silva 2012), as well as mathematical programming (Ülker and Landa-Silva 2010). More recent works include greedy algorithms with tabu search (Castillo, Riff and Montero 2016) and artificial bee colony algorithms (Bolaji, Michael and Shola 2017), that reported competitive results with the integer programming model from Ülker and Landa-Silva (2010). The heuristic algorithms developed until now for OSA tackle the problem as whole, and do not incorporate problem decomposition mechanisms. Consequently, the preferences of office occupants are not treated individually, but rather globally by aggregating them into the overall objective function of the problem.
Rather than developing more sophisticated heuristic algorithms, this study proposes a novel approach to solve the OSA problem by using the support of Agent Based Modelling (ABM). This allows the consideration of behavioral aspects and individual preferences of the office occupants. The remainder of the paper provides details about this approach to tackle the OSA problem and evaluates its performance by means of computational experiments on a set of benchmark instances from the literature.

2. BACKGROUND
2.1. Agent Based Modelling (ABM)
Modelling and simulation can be used to reproduce a real world environment with its features, forecast and explore different scenarios, experiment with possible alternative decisions, and set different values for decision variables. In ABM, a system is modelled as a collection of autonomous decision-making entities called agents (Pourdehnad, Maani and Sedehi 2002). Each agent individually assesses its situation and makes decisions on the basis of a set of rules. Individual agents interact with each other and their environment to produce complex collective behavior patterns. ABM is well suited to

ISBN 978-88-85741-03-4; Affenzeller, Bruzzone, Jiménez, Longo, Merkuryev and Piera Eds.

modelling systems with heterogeneous, autonomous and proactive actors, such as human-centered systems (Siebers et al. 2007).
ABM has a variety of applications from modelling the behavior of stock market and supply chains to predicting the spread of epidemics and understanding factors that may be responsible for the fall of ancient civilizations. All these make it possible to analyze the effects of changes in a specific system or environment, without affecting the real world during this process (Macal and North 2010).
Within the operations research discipline, agent based approaches have been used to solve problems like: scheduling (Sabar, Montreuil and Frayret 2011), knapsack (Polyakovskiy and M'Hallah 2007), transportation and supply chain planning (Persson et al. 2005). It has been reported that the agent based approaches proved to be competitive with heuristic optimization techniques, one advantage being that the problem can be divided into sub problems, and the work delegated to agents. Thus, when the size of the problem is large and the timescale is short, the decomposition nature of ABM can help to reduce the computational time needed to produce competitive solutions to the problem in hand (Barbati, Bruno and Genovese 2012). However, if the communication between agents has a high cost and if the high quality of solutions is a priority, then classical optimization techniques are preferred. According to Persson et al. (2005), who applied both ABM and classical optimization techniques to a production and transportation supply chain problem, a hybrid approach proved to be beneficial. They added optimization to agents during the decision making process, and obtained better results than having each method applied by its own. The performance of a hybrid system was also reported as highly satisfactory in a study by Daniels and Parsons (2006). They used agents to improve the population of candidate solutions in a genetic algorithm used to solve the zone-deck allocation of space in a preliminary ship general arrangements design.
According to Polyakovskiy and M'Hallah (2007), the agent based model developed for a knapsack problem, allows the investigation of a larger part of the solution space, leading to high quality solutions. They propose an agent based framework where the agent is characterized by its fitness, behavioral rules and parameters. The framework is then used in the decision making process to assess the potential of each possible action, and choose the one which optimizes the reward.
An interesting case of ABM is the Schelling Segregation Model (Schelling 1971), where, based on a simple rule of neighborhood satisfaction, people/entities of two types are moving on a grid and in time they start to segregate. Each entity is represented by an agent with an attached happiness value. If the percentage of agents of the same type among one's neighbors is lower than a certain threshold value, that agent changes the state to unhappy and moves to a randomly chosen new location. If the state remains happy, then the agent does not move. So, following a rule for changing the state of an agent, after several state changes, the agents exhibit patterns, order and structure in their behavior.
Starting from the principles of the Schelling Segregation Model and other examples of ABM for combinatorial optimization problems, this paper proposes an ABM approach for solving the OSA problem, where agents have an internal decision making process based on heuristics.

2.1.1. The Office Space Allocation Problem

The office space allocation problem used in this study was defined initially by Landa-Silva (2003) and updated later by Ülker (2013). This is a minimization combinatorial optimization problem that is highly constrained. In this problem there is a set of rooms – representing the space available for the allocation process, and a set of entities – representing people or resources that must be allocated to the rooms. Each room has a given size and each entity requires a given amount of space. There is also a set of constraints, which represent the requirements that should be met when creating an allocation. The constraints establish hard or soft restrictions on proximity and spatial relations among the entities and rooms. There are 9 types of such constraints: allocation, non-allocation, capacity, same-room, not-same-room, not-sharing, adjacency, nearby, away-from. The full definition and the penalty weights associated to each in the case of non-satisfaction are given in (Ülker 2013). A solution to the problem requires that each entity is allocated to a room. The quality of a solution is evaluated in terms of space misuse penalty (SMP in equation 1) and soft constraints penalty (SCP in equation 2). The aim is to minimize the total penalty (F in equation 3) associated with a solution. The usage of each room is calculated according to the entities allocated to the room. Overuse of space is less desirable hence penalized twice compared to underuse of space, as shown in equation 1. The violation of each constraint contributes with the associated weight towards the overall penalty for that solution.

$$SMP = \sum_{i=1}^{|R|} \max(cap(r_i) - usg(r_i), 2 \times (usg(r_i) - cap(r_i))) \quad (1)$$

$$SCP = \sum_{i=1}^{|C|} (w(c_i) \times v(c_i)) \quad (2)$$

$$F = SMP + SCP \quad (3)$$

In equations (1)-(3), $cap(r_i)$ is the capacity of room r_i, $usg(r_i)$ is the actual usage of room r_i, $v(c_i)$ is 1 if the constraint c_i is violated and 0 if it is satisfied, $w(c_i)$ is the weight of the soft constraint c_i, $|R|$ is the number of rooms in the problem, and $|C|$ is the number of constraints in the problem.
The data sets used in this paper were proposed by Ülker and Landa-Silva (2011). These problem instances vary on two dimensions: the number of constraints, and the space misused expected, all having a constant number of entities and rooms. One data set corresponds to a problem instance and it contains a list of entities (with id, group and space required), a list of rooms (with id, floor, and a

set of neighbors of the room) and a list of constraints (with id, type, subject and target – defined depending on the type of the constraint). These data sets are called SVe150 and PNe150 and were generated based on four parameters: S, V, P and N. For the parameters S, P and N, variations are made to adjust the size of the rooms as follows. S represents the slack space rate expected (general misuse of space), P represents a positive slack amount expected (underuse) and N represents a negative slack amount expected (overuse). The V parameter expresses the violation rate expected in regards to the constraints. The data sets selected for conducting experiments in this paper are: s000v000 - low space misuse and low constraint violation, s100v000 - high space misuse and low constraint violation, s000v100 - low space misuse and high constraint violation, s100v100 - high space misuse and high constraints violation, p025n000 - expect high underuse, p000n025 - expect high overuse.

3. METHODOLOGY

3.1. ABM for OSA

The agent based model developed here for OSA is a synchronous simulation model, and has the following components: entities, rooms, constraints and the environment. The *environment* is the space where all the agents are located and takes care of their coordination. It also keeps track of the global objective value for the current solution to the problem, using the information received from the rooms (about space misused) and from the constraints (about their violation). However, the environment does not have a mechanism to minimize this global objective, but rather it monitors the objective value and the solution, making the calculations available for agents on request. The responsibility for minimizing the objective function value is distributed to the agents of the system. The rooms and the constraints are represented as passive agents (objects). Entities (people) are represented as active agents (entities from OSA problem definition). From this point onwards in this paper, active agents are referred to as *agents* and passive agents as *objects*.

Being a synchronous model, all agents move simultaneously, evaluating possible rooms for themselves. The system then gathers all the individual allocations of agents to rooms and builds the overall solution for the problem. Because of these mechanisms, this approach can be seen as distributed optimization. The overall evaluation function is split based on constraints. Then, some components of that evaluation function are delegated to each individual agent.

Figure 1 presents the conceptual model of the system, showing the interaction between the different components. Agents, through their behavior, search and build a solution for the problem. They have two possible states, *satisfied* and *not satisfied* and they move between them based on an internal evaluation process. This will be detailed in the next subsection. During their internal evaluation process, the agents request information from the rooms and from the constraints. From the constraints, the agents get a list of those directly involving them and in turn affect their satisfaction score. From the rooms, the agents get information about occupancy of the rooms and the neighborhood relationships between rooms defined by the problem instance.

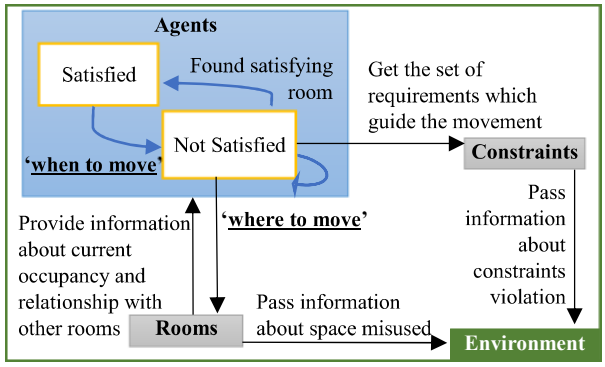

Figure 1: Agent Based Model for the Office Space Allocation Problem.

3.2. Agents Structure

As mentioned in the previous section, the agents proposed in this model represent the entities in an instance of the OSA problem. During their search process, agents move between rooms and analyze their own satisfaction state, choosing 'when' and 'where' to change their allocation. The state of an agent can be either *satisfied* or *not satisfied*. Four strategies are proposed for agents to decide 'when to move':

- Percentage Based (PB): The agent has a current penalty given by the constraints which are associated with it and are violated in the current position (calculations of this penalty are detailed in Algorithm 1). If this penalty is more than 30% of the total possible penalty, then the agent moves. The total possible penalty is calculated as a pessimistic scenario, when all the constraints associated with the agent are violated, thus is the maximum penalty that the agent can have.
- Always Move (AM): All the agents are moving at every step of the simulation.
- Highest Penalised (HP): The agent with a current penalty larger than the average penalty for all agents in that particular moment, are moving to a different location. This implies that every step approximately half of the agents are moving to a different room.
- Random Chance (RC): Each agent has a chance of 1 in 6 to move to a different location. This chance is based on the 'dice rolling' probability.

Depending on the outcome of the strategy applied, an agent makes the first decision from its internal 'thinking process', namely it defines its satisfaction state. If this process results in being *not satisfied* and therefore having to move to a different room, the agent goes to the second stage, and it analyses 'where to move'. This means that the agent searches for a room which will fulfil its

ISBN 978-88-85741-03-4; Affenzeller, Bruzzone, Jiménez, Longo, Merkuryev and Piera Eds.

requirements and get back to a *satisfied* state. If a room is not found, or it does not improve its current position, then the agent stays in the current room returning to a *not satisfied* state.

Algorithm 1 Penalty calculation for an agent

```
agent.penalty = 0
for (constraint c : agent.associatedConstraints)
    switch (c.type)
        if(c is violated) //defined based on c.type
            agent.penalty += c.weight
        end if
agent.penalty += agent.room.misusedPenalty /
                 room.noOfAgentsInRoom
return agents.penalty
```

There are four strategies proposed for the 'where to move' decision. These are heuristic based approaches, and vary from random to greedier ones:

- Rnd: The choice of the new room for the agent is picked at random.
- BestPotential: The agent evaluates the potential of each available room and selects the one which returns the smallest penalty in the scenario of moving there. The potential of a penalty is relative to other agents moving in the same time, thus the evaluation may not be the same before and after the move is done.
- AllConstr: The agent is trying to find a room which will not violate its associated constraints. Instead of analysing all the rooms and calculating a penalty, this approach is narrowing down the list of rooms based on the constraints. If there is no room satisfying all constraints, the agent goes back and tries to satisfy most of the constraints.
- OneByOne: At every simulation step, the agent picks at random currently violated constraint and finds a room which will satisfy it in the scenario of moving there.

The simulation based ABM executes the search process in steps. At each step, all the agents act simultaneously. Before the step, agents calculate their satisfaction, based on one of the four strategies mentioned. Then, they choose the room where to move, again using one of the strategies mentioned. On the actual step, the agents are essentially executing the move, so they change their allocation to a new room. The entire process is represented in Algorithm 2.

Algorithm 2 ABM simulation process

```
On start-up:
    initialize the agents from external database
    build the neighborhood links between rooms
    assign the constraints to the agents (entities)
On before step:
    agents.satisfied = agents.decide('when to move')
    if (agents.satisfied = false)
        newRoom = agents.choose('where to move')
On step:
    if (agents.satisfied = false)
        agents.room = newRoom
On after step:
    rooms.misusedPenalty -> update (with new
    occupancy)
    agents.totalPenalty -> update (in the new allocation)
    environment.fitnessFunction ->update (from agents)
```

The ABM for OSA described above is relatively simple but is it designed to serve as a baseline methodology to investigate the suitability of tackling OSA through a distributed optimization paradigm instead of a typical centralized optimization or heuristic approach. This investigation will hopefully inform the development of more elaborate behavior in the agents and conditions in the environment.

4. EXPERIMENTAL RESULTS

There are 16 behaviors which can be exhibited by an agent, combining the two decision mechanisms of 'when' and 'where' to move. This resulted in 16 different neighborhood exploration strategies that are evaluated. The simulation was run with each algorithm for 5000 steps, and 50 replications on each data set. The number of steps was set to 5000, because preliminary experiments showed that after this point all the agents' strategies did not continue to improve the overall solution for the problem. The results presented in Table 1 correspond to the s000v000 problem instance. The same rankings of the algorithms was also observed in the other problem instances mentioned in section 2.1.1 above. For this reason, details of the experimental results for those other instances are omitted here but are available for further analysis.

Table 1: Neighbourhood exploration strategies built in the ABM for OSA on the s000v000 problem instance

ABM Strategy	Diversity	Min	Mean	Max
PB-Rnd	50.08	2878.00	3181.68	3553.50
PB-BestPotential	0	3971.50	3971.50	3971.50
PB-AllConstr	30.96	1969.00	2432.36	2808.00
PB-OneByOne	33.36	1691.50	2247.53	2710.00
AM-Rnd	70.78	2890.00	3591.48	4076.00
AM-BestPotential	0	8238.50	8238.50	8238.50
AM-AllConstr	34.69	1853.50	2602.69	3150.00
AM-OneByOne	32.94	1835.00	2499.79	3083.50
HP-Rnd	52.25	2942.50	3444.55	3871.50
HP-BestPotential	0	4210.00	4210.00	4210.00
HP-AllConstr	27.33	1889.50	2465.24	2991.00
HP-OneByOne	31.24	1751.00	2279.42	2761.50
RC-Rnd	66.31	2347.50	2599.97	2952.50
RC-BestPotential	41.54	3053.00	3502.37	4265.50
RC-AllConstr	21.29	1536.00	1899.82	2318.00
RC-OneByOne	21.40	1346.00	1751.50	2069.00

ISBN 978-88-85741-03-4; Affenzeller, Bruzzone, Jiménez, Longo, Merkuryev and Piera Eds.

Table 1 contains the min (best), mean and max (worst) evaluation function values for the best solutions produced by each combination of agents' strategies. The diversity value represents the degree of difference in the actual structure of best solutions produced. A detailed description of how the diversity is calculated can be found in the next section.

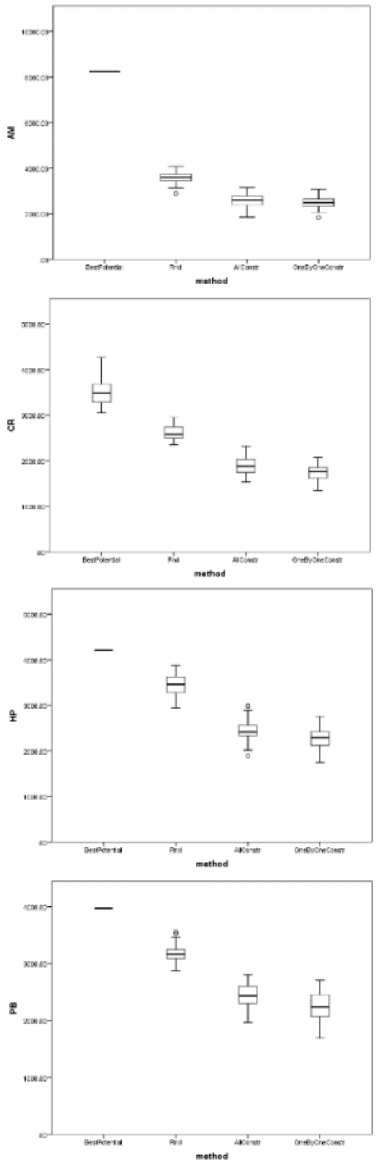

Figure 2: Boxplots of 'where to move' strategies in all the 'when to move' cases.

The results presented in Table 1 show that, in general, the RC strategy performs best amongst 'when to move' strategies, and OneByOne approach performs best amongst 'where to move' strategies. This can also be seen in the boxplots from Figure 2. The rankings of the 'where to move' strategies are maintained throughout all the 4 'when to move' cases. From best to worse, the ranking of the algorithms is: OneByOne, AllConstr, Rnd and BestPotential. Contrary to what it name suggests, the BestPotential approach performs very poor, because it is heavily relying on the penalty evaluation done 'on before step', which does not correspond with the actual penalty seen 'on after step'. RC, although being of a random nature, it is the 'when to move' strategy that makes the least amount of agents to move simultaneously, thus the least disruption in the environment from one step to another. Therefore, RC is the best performing strategy because the penalty evaluations made by an agent are less likely to change from 'on before step' to 'on after step'.

4.1. Diversity in the solutions

The diversity in the solutions produced by one algorithm is represented in this paper as the degree of differences in the structure of the best solutions returned throughout all the repetitions of the experiments. This is calculated using equation (4) taken from (Landa-Silva 2003). The diversity ranges from 0 to 100, 0 meaning that all the solutions compared are exactly the same and 100 meaning that all the solutions compared do not have any one element in common. A solution of the office space allocation problem is represented as an array, where the index represents the entity (agent) id and the value in the array represents the room id to which that agent is allocated (see Figure 3). Having 50 such arrays (solutions), from the 50 replications of an experiment, the diversity is represented by the agent being allocated to different rooms in those solutions. So, if the value in the solution array corresponding to and agent is different from the value in the same position in another solution array, then the two solutions have a difference between them in that position. D(j) is the number of unique values in the j^{th} position of the solution, p is the number of solutions compared, and n is the length of the solution. Figure 4 illustrates the use of this equation for p = 5 solutions and n = 7 agents.

$$V(p) = \frac{\sum_{j=1}^{n} \frac{D(j)-1}{p-1}}{n} \times 100 \qquad (4)$$

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	...	$e_{\|E\|}$
r_5	r_7	r_1	r_9	r_2	r_5	r_3	r_8	...	r_6

Figure 3: Representation of a solution for OSA.

	A	A	A	A	A	A	A
	A	A	B	B	A	B	B
	A	B	B	C	B	C	C
	A	B	B	C	B	D	D
	A	B	B	C	C	D	E
D(j)	1	2	2	3	3	4	5
(D(j) − 1) / (p − 1)	0	0.25	0.25	0.50	0.50	0.75	1

V(p) = (3.25 / 7) x 100 = 46.42 %

Figure 4: Example of calculation of the diversity in solutions.

Looking at the diversity column in the results presented in the previous section, it can be observed that the Rnd strategy scores the highest. It is expected to have a higher diversity in the solutions, when rooms are chosen at random, and therefore the agents explore more. On the other hand, BestPotential strategy, being a greedy one, almost in all cases chooses the exact same rooms, and

therefore producing the same solutions regardless of the number of the experimental repetitions. In general, the RC strategy also scores lower in terms of diversity, because a smaller number of agents move at a time, thus creating a smaller disturbance in the solutions.

5. COMPARISON

5.1. Heuristics for Neighborhood Exploration

For comparison purposes, the ABM strategies for neighborhood exploration are evaluated against known heuristic algorithms used for neighborhood exploration on the OSA problem. They were originally proposed by Landa-Silva (2003) and applied to the OSA data sets from three UK universities. This paper presents a replication of those heuristics and two additional ones, implemented for this study, all tested on the new generated data sets by Ülker and Landa-Silva (2013). These heuristics are based on the main idea of the relocation of an entity to a different room. The methods of selecting the entity to relocate vary from completely random to greedier ones:

- RelocateRndRnd: Selects an entity at random and relocates it to another random room;
- RelocateRndBestRnd: Selects an entity at random and relocates it to the best room, in terms of evaluation function value, out of a randomly selected subset of rooms;
- RelocateRndBestAll: Selects an entity at random and relocates it to the best room out of all the available rooms;
- RelocatePntyBestRnd: Selects the entity with the highest current penalty associated with it and relocates it to the best room out of a randomly selected subset of rooms;
- RelocatePntyBestAll: Selects the entity with the highest current penalty and relocates it to the best room out of all the available rooms.

These heuristics are tested separately using an 'only improving' local search algorithm (see Algorithm 3). The reason for applying each neighborhood exploration strategy separately, and using a simple algorithm, is to analyze the individual behavior and identify the strengths and weaknesses, without having external influencing factors.

Algorithm 3 Local search

```
generate initial solution X
repeat
    apply neighborhood heuristic (select entity and room
     for the move)
    apply the relocation move to X to produce X'
    perform acceptance test of X'
    if (X' is better than X)
        X = X'
until termination criteria
```

Each neighborhood exploration heuristic was run until no improvement was found for a number of iterations. Each experiment was replicated 50 times as it was the case for the ABM simulation. The results are included in Table 2.

Table 2: Heuristic neighborhood exploration algorithms for OSA on the s000v000 problem instance

Heuristic	Diversity	Min	Mean	Max
RndRnd	52.34	1558.00	1983.27	2439.00
RndBestRnd	52.67	1446.00	1930.56	2361.00
RndBestAll	50.89	1364.50	1861.27	2383.00
PntyBestRnd	51.77	1785.50	2348.92	2818.00
PntyBestAll	52.60	1493.00	2304.80	2800.50

The differences between the heuristic neighborhood exploration algorithms seen in the boxplots (Figure 5) are not major. However, they do show that RndRnd, RndBestRnd and RndBestAll are overall better than the PntyBestRnd and PntyBestAll. Performing ANOVA statistical analysis confirmed that the RndBestAll is the best performing algorithm when compared with the rest of the heuristics, having a significance difference p value of less than 0.005.

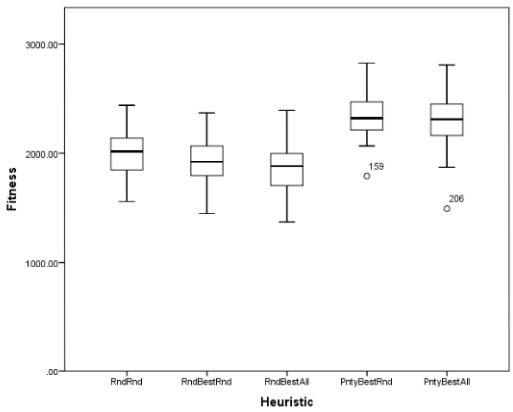

Figure 5: Boxplots for all the heuristic neighborhood exploration algorithms on s000v000.

5.2. Comparison of the results

The two winning neighborhood exploration strategies, namely the heuristic RndBestAll and the ABM CR-OneByOne, are compared with each other on all the 6 problem instances selected: s000v000, s000,v100, s100v000, s100v100, p025n000, and p000n025. The results are included in Table 3. For all the instances except p025n000, there is statistical significance between the two methods. CR-OneByOne is the one which produces better results for almost all the instances in terms of evaluation function value of a solution for OSA. The exceptions are for the s100v100 and p025n000 instances. For p025n000 the statistical analysis does not show significant difference between the two algorithms. However, for s100v100 the difference is statistically significant, RndBestAll being the better one. The s100v100 problem instance is expected to have high space misuse and high constraints violation. This might

ISBN 978-88-85741-03-4; Affenzeller, Bruzzone, Jiménez, Longo, Merkuryev and Piera Eds.

suggest that the current ABM simulation for OSA is not coping well with high expected penalty, both from constraints and from space misused. However, this cannot be generalized by looking at only one instance of this type for the problem. Further analysis has to be made in order to confirm this conclusion.
In terms of the diversity of the solutions produced by the two algorithms, RndBestAll has a higher score (50-55%), while CR-OneByOne has a lower one (20-30%). The ABM CR-OneByOne produces less diversity in the solutions because the agents have the option of not moving if they do not find a promising better room than the one they already are allocated to, hence not exploring more rooms.

Table 3: Comparison of RndBestAll and CR-OneByOne algorithms for OSA on the s000v000, s000_v100, s100v000, s100v100, p025n000 and p000n025 problem instances

Algorithm	Diversity	Min	Mean	Max
p000n025				
CR-OneByOne	21.43	1541.80	1901.66	2291.20
RndBestAll	52.15	1698.00	2107.18	2662.20
p025n000				
CR-OneByOne	23.95	1496.00	1938.04	2359.10
RndBestAll	51.54	1418.60	1889.23	2339.30
s000v000				
CR-OneByOne	21.40	1346.00	1751.50	2069.00
RndBestAll	50.89	1364.50	1861.27	2383.00
s000v100				
CR-OneByOne	25.42	1539.00	2035.58	2522.50
RndBestAll	51.21	1723.50	2103.48	2503.00
s100v000				
CR-OneByOne	17.61	1536.80	1913.06	2314.60
RndBestAll	54.32	1560.90	2023.64	2498.20
s100v100				
CR-OneByOne	27.31	1810.70	2282.53	2564.40
RndBestAll	54.11	1653.10	2162.81	2606.90

The performance of the neighborhood exploration strategies is given by the overall value of the objective function. However, for a better understanding of the strengths and weaknesses of the approaches used to solve the OSA problem, the objective function values obtained were split by constraints and space misuse penalty. As it can be seen in Figure 6 and Figure 7, the ABM CR-OneByOne strategy is optimizing better the satisfaction of constraints, having less violation than the RndBestAll heuristic. On the other hand, the RndBestAll strategy is coping better with the space misuse, reporting lower values for this component of the objective function. The agents built in the simulation model do not have mechanisms to cope with space misuse, so there is potential for improvement in this direction.

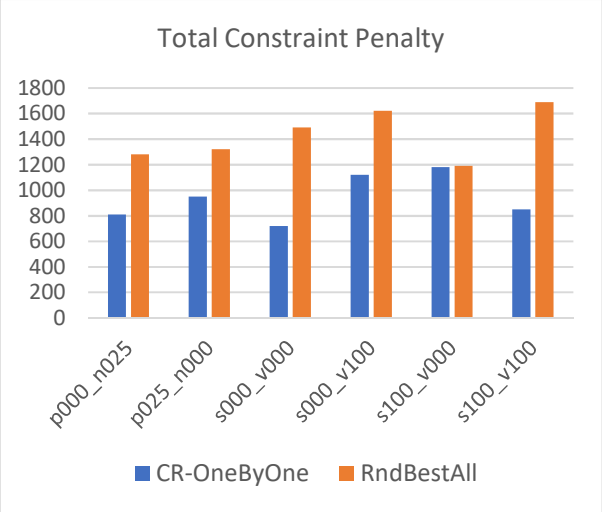

Figure 6: Constraints penalty obtained by the ABM and heuristic neighbourhood strategies for all the instances.

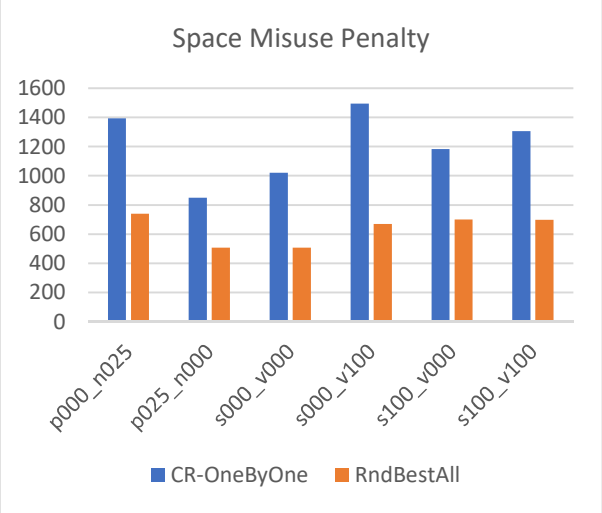

Figure 7: Space misuse penalty obtained by the ABM and heuristic neighbourhood strategies for all the instances.

Going further and splitting the total constraint penalty by each of the individual constraints, it is observed that CR-OneByOne get the highest penalty from 'nearby', 'not sharing' and 'same room' constraints (see Figure 8). The heuristic for neighbourhood exploration, RndBestAll, gets the highest penalty from 'nearby', 'allocation' and 'same room' (see Figure 9).
From the analysis, the overall performance of the strategies compared in this paper does not seem to depend much on the problem instance. The constraint penalty (total or split by individual constraints) and space misuse penalty keep their proportions across all the instances.

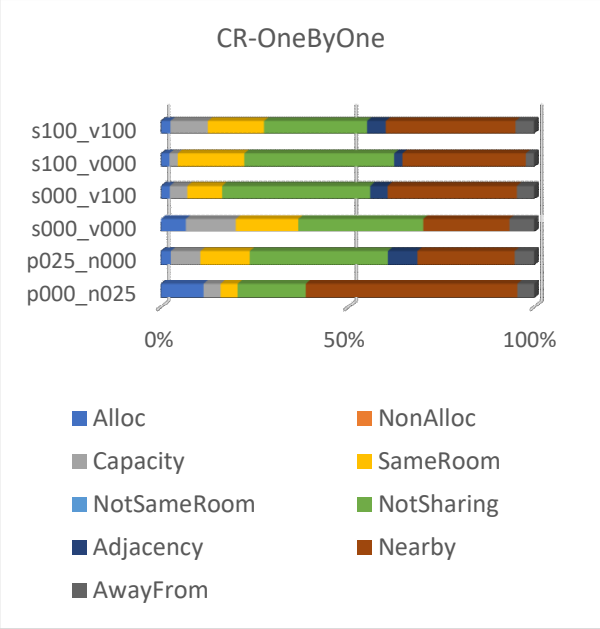

Figure 8: Individual constraint penalty obtained by CR-OneByOne for all the instances.

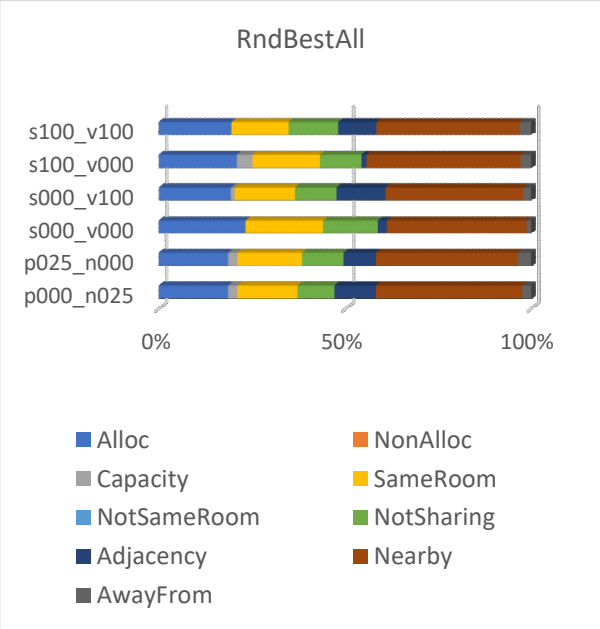

Figure 9: Individual constraint penalty obtained by RndBestAll for all the instances.

6. CONCLUSION

6.1. Discussion

A key factor which influences the performance of the behaviors in the proposed ABM is the fact that the model is implemented synchronously. This means that agents move in steps, and at the same time with others, as described in this paper. So the environment constantly changes, and the calculations that an agent does when deciding 'where to move', might be completely wrong, because they depend on other agents and their locations too. Thus, when one moves, it can negatively influence the decisions which were already taken by another agent.

The BestPotential strategy is the most appropriate example in this case, when the synchronous nature of the system influences a moving strategy to the point where it makes it become the worst performing strategy. OneByOne mechanism is the least affected by the synchronous nature of the system, and in consequence is the best performing algorithm from ABM.

From the 'when to move' strategies, the RC performs better because it is less influenced by the fact that the environment changes. On the other and, the AM strategy is among the worst in terms of performance, because the calculation of potential penalty does not match the actual penalty once the move is made.

A solution to this problem would be to create an asynchronous system or to include communication mechanisms between agents. This way agents can still follow their individual goal, but also collaborate with each other in taking decisions and coordinate their movement when they are involved in the same constraints. The results in the previous section about the individual contribution of the different types of constraints to the overall penalty from the objective function, indicate that the ABM can be further improved to better cope with the 'nearby', 'not sharing' and 'same room' constraints. This is a promising direction to further develop the ABM in the future.

Regarding the heuristics, it was seen that PntyBestRnd and PntyBestAll are not performing better than the rest of the methods, although they are always trying to move the highest penalized entity and find a better allocation for it. This is because these heuristics do not explore too much the search space and get very easily trapped in a local optima. The heuristics considered here move from a solution to another by changing only one entity from a current allocation to another, limiting the neighborhood exploration, which also influences the performance of the algorithms.

A future direction for this study is to analyze all the other available instances for the problem, and see if the performance of the agent based model is still comparable to the neighborhood search heuristics. A couple of improvements to the existing agent based model can also be made. For example, adding a learning mechanism to the agents, and including communication between agents as mentioned previously. Although the agents in this model are working and following an individual goal, they have the potential of improving their movement strategies by collaborating with each other. The constraints, due to their nature, usually involve more than one entity (agent) in their formulation. Therefore, negotiation mechanisms between the agents involved in the same constraint will make them work towards the same goal, and possibly improving on the results presented in this paper.

6.2. Summary

This paper presented an agent based simulation model (ABM) for the OSA problem that overall performs better than or as good as the heuristics used as neighborhood exploration in the literature. The model is based on the

individual agents' behavior and strategies, which aim to improve their current position. This approach allowed for problem decomposition to be achieved, which is also the first time this was done for the OSA problem.

There is room for improvement in the model presented, and the fact that it already produces results comparable with existing methods, suggests that it is a promising direction for future research in this problem.

OSA can be associated with a general assignment problem, it has similarities with bin packing, container loading and general resource allocation. It has many more types of constraints than these well-known problems, which is making it more difficult to solve. However, the model used for solving OSA can be also applied to these other similar assignment problems.

ABM simulation proves to be a good methodology for solving the office space allocation problem, thus making this a valuable contribution to the field, because the same principles are easy to adapt to other real world assignment problems.

REFERENCES

Adewumi, A. O., and Ali, M. M., 2010. A multi-level genetic algorithm for a multi-stage space allocation problem. Mathematical and Computer Modelling, 51(1-2), 109-126.

Awadallah, M. A., Khader, A. T., Al-Betar, M. A., and Woon, P. C., 2012. Office-space-allocation problem using harmony search algorithm. In International Conference on Neural Information Processing (pp. 365-374). Springer, Berlin, Heidelberg.

Barbati, M., Bruno, G., and Genovese, A., 2012. Applications of agent-based models for optimization problems: A literature review. Expert Systems with Applications, 39(5), 6020-6028.

Bolaji, A. L. A., Michael, I., and Shola, P. B., 2017. Optimization of Office-Space Allocation Problem Using Artificial Bee Colony Algorithm. In International Conference in Swarm Intelligence (pp. 337-346). Springer, Cham.

Castillo, F., Riff, M. C., and Montero, E., 2016. New Bounds for Office Space Allocation using Tabu Search. In Proceedings of the 2016 on Genetic and Evolutionary Computation Conference (pp. 869-876). ACM.

Daniels, A., and Parsons, M. G., 2006. An agent based approach to space allocation in general arrangements. Proceedings of the 9th IMDC, Ann Arbor, MI.

Kincaid, R., Gates, R., and Gage, R., 2007. Space allocation optimization at nasa langley research center. In Proceedings of the Seventh Metaheuristics International Conference, Montreal, Canada.

Landa Silva, J. D., 2003. Metaheuristic and multiobjective approaches for space allocation. Thesis (PhD). The University of Nottingham.

Landa-Silva, D., and Burke, E. K., 2007. Asynchronous cooperative local search for the office-space-allocation problem. INFORMS Journal on Computing, 19(4), 575-587.

Lopes, R., and Girimonte, D., 2010. The office-space-allocation problem in strongly hierarchized organizations. In European Conference on Evolutionary Computation in Combinatorial Optimization (pp. 143-153). Springer, Berlin, Heidelberg.

Macal, C. M., and North, M. J., 2010. Tutorial on agent-based modelling and simulation. Journal of simulation, 4(3), 151-162.

Persson, J. A., Davidsson, P., Johansson, S. J., and Wernstedt, F., 2005. Combining agent-based approaches and classical optimization techniques. In Third European Workshop on Multi-Agent Systems.

Polyakovsky, S., and M'Hallah, R., 2007. An agent-based approach to knapsack optimization problems. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 1098-1107). Springer, Berlin, Heidelberg.

Pourdehnad, J., Maani, K. and Sedehi, H., 2002. System dynamics and intelligent agent-based simulation: where is the synergy. In Proceedings of the 20 International Conference of the System Dynamics Society.

Sabar, M., Montreuil, B., & Frayret, J. M., 2009. A multi-agent-based approach for personnel scheduling in assembly centers. Engineering Applications of Artificial Intelligence, 22(7), 1080-1088.

Schelling, T. C., 1971. Dynamic models of segregation. Journal of mathematical sociology, 1(2), 143-186.

Siebers PO, Aickelin U, Celia H and and Clegg C, 2007. A Multi-Agent Simulation of Retail Management Practices. In: Proceedings of the 2007 Summer Computer Simulation Conference (pp. 959-966), San Diego, CA, USA.

Trung, N. T., Tuan, T. N., and Anh, D. T., 2009. Informed simulated annealing for optimizing dorm room assignments. In Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on (pp. 265-270). IEEE.

Ülker, Ö., and Landa-Silva, D., 2010. A 0/1 integer programming model for the office space allocation problem. Electronic Notes in Discrete Mathematics, 36, 575-582.

Ülker, Ö., and Landa-Silva, D., 2011. Designing difficult office space allocation problem instances with mathematical programming. In International Symposium on Experimental Algorithms (pp. 280-291). Springer, Berlin, Heidelberg.

Ülker, Ö., and Landa-Silva, D., 2012. Evolutionary local search for solving the office space allocation problem. In Evolutionary Computation (CEC), 2012 IEEE Congress on (pp. 1-8). IEEE.

ISBN 978-88-85741-03-4; Affenzeller, Bruzzone, Jiménez, Longo, Merkuryev and Piera Eds.

Ülker, Ö., 2013. Office space allocation by using mathematical programming and meta-heuristics. Thesis (PhD). The University of Nottingham.

AUTHORS BIOGRAPHY

Alexandra Cristina Dediu is a PhD student from the Automated Research Optimization and Planning (ASAP) Research Group, in the School of Computer Science, University of Nottingham, UK. Her main research interests are in the optimization of problems related to resource allocation, especially when solutions can improve the businesses processes and facilitate management operations.

Dr Dario Landa-Silva is an Associate Professor in the School of Computer Science at the University of Nottingham in the UK. His research interests are in the interface between computer science, operations research and artificial intelligence for the application of modeling, search and optimization techniques to underpin the development of intelligent decision support systems across a range of real-world applications, particularly in logistic and operational scenarios. More recently, he is also conducting research on the interplay between optimization and other methodologies such as machine learning, computer simulation, data science and human computation. He has over 95 scientific publications in refereed journals, edited books, and proceedings of international conferences and workshops.

Dr Peer-Olaf Siebers is an Assistant Professor at the School of Computer Science, University of Nottingham, UK. His main research interest is the application of computer simulation to study human-centric complex adaptive systems. He is a strong advocate of Object Oriented Agent-Based Social Simulation. This is a novel and highly interdisciplinary research field, involving disciplines like Social Science, Economics, Psychology, Operations Research, Geography, and Computer Science. His current research focusses on Urban Sustainability and he is a co-investigator in several related projects and a member of the university's "Sustainable and Resilient Cities" Research Priority Area management team.