A COMPUTER BASED SYSTEM FOR SPACE ALLOCATION OPTIMISATION

E.K. Burke, P. Cowling, J.D. Landa Silva, B. McCollum, D. Varley
Automated Scheduling, Optimisation and Planning Group
Department of Computer Science, University of Nottingham, UK

Abstract. In many institutions of higher education, office space is a precious commodity and its correct utilisation affects the functioning of the working environment. In academic institutions in particular, there are continuous changes in rooms and/or resources that have a direct impact on the space distribution. Therefore, improving the existing distribution of rooms is a continuous and extremely important process for estates management officers. Based on the use of advanced searching techniques we present a computer system to automate this multi-objective and highly constrained process, specifically in academic institutions. Our system is designed for 32 bit Windows environments, provides database support and graphic output through ODBC and OLE respectively, lets the user specify particular requirements and constraints for the problem, produces statistical information on space utilisation and unsatisfied constraints, shows room and resource distribution and permits manual changes in the allocation.

Key words. space allocation, optimisation, heuristics, hill climbing.

1 INTRODUCTION

Space management can be carried out more efficiently when the building design process has been thoroughly planned. Approaches based on linear goal programming [2] and more recently, ant colony optimisation [3] have been proposed to design academic and commercial facilities. However, when the usable space is already constructed, the problem of space allocation and the automation of this process is very important in different areas. A dynamic programming model to determine the shelf-space needed for products in a supermarket was presented in [14]. In [11], the allocation of storage space to inventories was tackled using a transportation problem model, while a specific heuristic was designed in [10] to assign space and the necessary number of transfer cranes for import containers in sea ports. In academic facilities, some approaches using linear goal programming have been proposed as in [8], [9] and [13]. Space allocation refers to the problem of distributing the available areas of space to particular "objects" each with different size requirements, so as to ensure satisfaction of certain constraints and to try to satisfy as many desirable (but no essential) constraints as possible. In higher education institutions, these "objects" can be staff, students, lectures rooms, laboratories, special rooms, etc., while areas of space are the rooms that can be used to allocate these resources. In [4] it was demonstrated that in UK universities, space allocation is a very important and complicated problem, which is highly constrained, has multiple objectives and varies greatly from one institution to another.

The optimisation of space allocation in universities is defined as the process of improving the existing space allocation, by means of reassigning resources and satisfying as many requirements and constraints as possible. An example of requirements is the necessary space for each resource, while constraints are specific restrictions that should or must be fulfilled within the particular scenario. The types of constraints considered include: proximity/adjacency requirements, sharing restrictions, grouping requirements, space requirements, limits for space wastage and space overuse, requirements concerning shared staff between different departments and resources that must be placed in specific locations. Attempting to improve the current distribution of rooms in academic institutions can be necessary for several reasons: the availability of rooms is modified, addition of new staff or students, special resources are reallocated, rooms are resized, removal of staff or students, changes in space requirements, addition/removal/change of constraints, or simply, as an attempt to increase efficiency. In this paper, we present a computer system which is flexible enough to be applied to the optimisation process in different situations. We also present results obtained when our system was tested using real data. The proposed system and the techniques implemented, can be transferred to a variety of industrial and commercial organisations with similar space optimisation problems.

2 SYSTEM FEATURES

The Automated Space Allocation System presented in this paper, has been developed as a result of research carried out within the Automatic Scheduling, Optimisation and Planning group in the University of Nottingham [5]. A questionnaire was sent to ninety-six universities in the UK as part of this project, and from this survey we established a collection of minimal requirements of a system to automate the space allocation process [4]. This collection can be summarised as follows:

- **Hardware Compatibility:** The system has been designed to run under PCs running 32 bit Windows environments such as Windows 95/98 or NT because this is the configuration that most British universities use.
- **Connectivity:** The integrated ODBC support provides a straightforward manner in which to load the necessary database information to initiate the optimisation process. Using ODBC libraries we ensure that the system is able to read the user's data in standard formats such as SQL, Oracle, Access, Excel, Text CSV and others.
- **Visualisation:** The system has the ability to display all available information regarding the current optimisation problem. Fig. 1 shows the *fitness statistics* window that displays the number of allocated resources, the number of rooms used, space utilisation and penalties for unsatisfied constraints. Fig. 2 displays the *resource summary* window that presents a list of each resource together with its corresponding assigned room, constraint penalties and other details from the database.

Fig. 1 Fitness statistics window

Fig. 2 Resources summary window

Fig. 3 shows the *room summary* that displays a list of all rooms in the problem and for each room, the resources that are allocated to it, the space utilisation and details from the loaded database. There is also an *unallocated resources* window that lists those resources that were not allocated to any individual room and a *room changes* window, where the user can see the list of those resources that were allocated or moved from one room to another since the data was first loaded into the system. With the added OLE support, the system can display layouts of the space that is being used for the current optimisation problem as well as graphical representation of the fitness statistics.

- **Ease of use:** Space administrators can easily use our software thanks to the user-friendly interface, enabling them to configure the system as well as to interpret, modify and save the solutions produced. A file set containing the required database information for the optimisation problem must be loaded into the system. This file set splits the database into three groups: resources, rooms and constraints. *Resources* is the list of all resources to be considered in the optimisation problem together with appropriate information such as name, level, owner, group, quantity, share, priority, space requirement and use. *Rooms* contains the available rooms to be used in the allocation problem together with the required data for each particular room such as label, size, resource, building, floor, adjacent rooms,

owner, type and use. *Constraints* lists all standard and problem specific constraints to be used in the optimisation process. The specification and modification of constraints can be controlled using a specially designed window through which the user is able to define label, constraint, subject, target, type, weighting and priority for each constraint. Once the system is loaded with all the required data, there are two user levels to configure the system. The first level, *system administrator*, shown in Fig. 4, authorizes the choice of an appropriate algorithm and its parameters, while the second level, *system user*, decides between a quick and a more thorough but slower searching process. Once the system produces the solution, the space manager can evaluate it using the displayed information. It is also possible to modify this proposed solution through added editing capabilities that let the user move resources from one room to another, obtaining information from the system about the evaluation of this move.

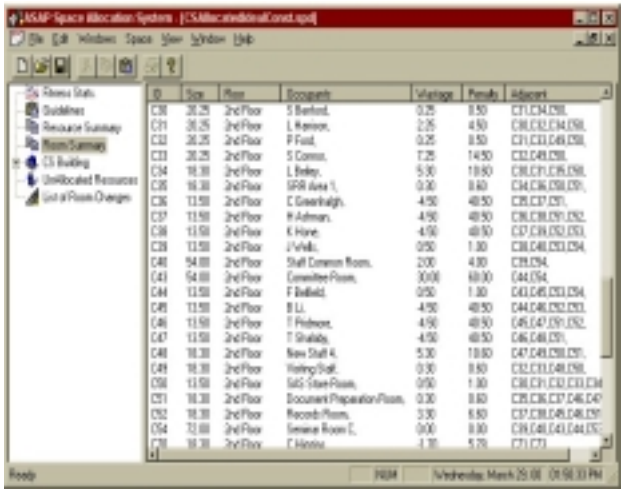

Fig. 3 Rooms summary window

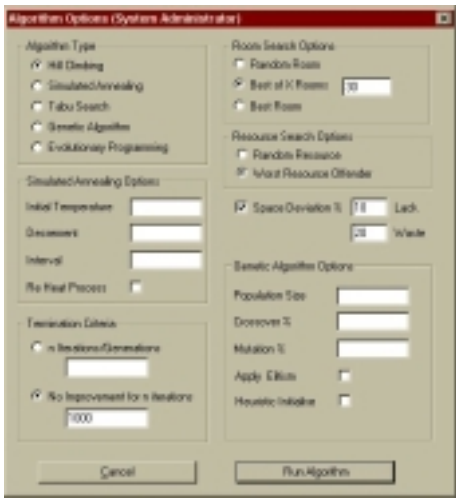

Fig. 4 Administrator window

- **Functionality:** We are providing space managers with a very functional tool which enables them to evaluate, according to their specific environment, the existing space allocation. It is possible to find different solutions focused upon specific goals by specifying the importance of each constraint. The system can be configured to use the techniques implemented with different parameter values that also take into account these specific user goals. In this way, different solutions can be proposed for the same problem depending on the set of constraints, and the algorithm settings.

3 AUTOMATING SPACE USAGE OPTIMISATION

3.1 The Allocation structure

The arrangement used to represent an allocation or solution in our system, is based on three main data structures that are described as follows:

ResourceGene is the structure that contains one resource's data, its fitness value, the assigned room, a pointer to the next ResourceGene sharing the same room and a pointer to the first ConstraintGene that is applied to this resource.

RoomGene incorporates one room's data and its fitness statistics. These fitness statistics are: the amount of space wasted, the amount of space overused, penalty due to wastage, penalty due to overuse, penalty due to resource conflicts (constraint violations), the total fitness value, a pointer to the first ConstraintGene that is applied to this room and a pointer to the first ResourceGene that is allocated in this room.

ConstraintGene is the structure that represents each particular constraint in the current problem and is composed of the penalty value applied when this constraint is not satisfied and a pointer to the next ConstraintGene that is applied to the same resource or room.

An example of an allocation representation is given in Fig. 5, where we can observe that there are 5 resources, 3 rooms and 3 constraints. Resource 1 is allocated to room III, resources 2, 4 and 5 are

allocated to room I, resource 3 is not allocated and room II is empty. Constraint A applies to room I, while constraints B and C apply to resource 2.

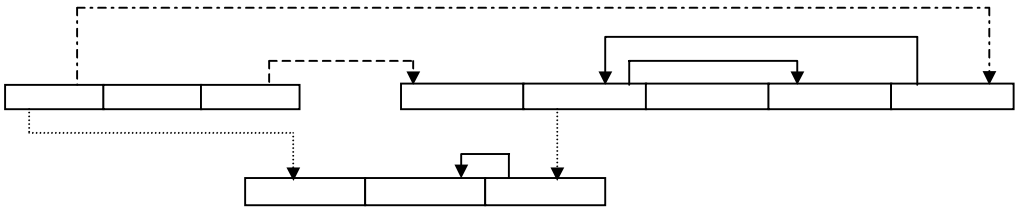

Fig. 5. The allocation structure

3.2 Evaluation

We evaluate solution quality using the following penalty function:

$$penalty = \sum_{i=1}^{N} UP(r_i) + \sum_{i=1}^{M} [WP(s_i) + LP(s_i)] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} RCP(r_i, r_j) \quad (1)$$

where UP is the penalty applied to the resource r_i if it was not scheduled, WP is the penalty applied to the room s_i if there is space wastage, LP is the penalty applied to the room s_i if there is space overuse, RCP is the penalty if there is a conflict (constraint violation) between resources r_i and r_j, N is the total number of resources to be allocated in the problem and M is the total number of rooms to be used in the allocation process.

3.3 The Implemented Algorithm

Several techniques have been implemented and tested in our system for the space allocation problem and new approaches are being investigated, additional information can be found on the web [6]. When attempting to optimise an existing allocation, our Hill-Climbing algorithm is the one that produces the best results. Hill-Climbing is well known as a local search strategy that attempts to optimise the current solution by means of progressive improvements to the current solution [1], [12]. The heuristic search that we incorporated into the Hill-Climbing technique is based on three possible operations: ALLOCATE (which finds a room to allocate an unscheduled resource, if this exists), MOVE (where one resource is moved from one room to another) and SWAP (which interchanges the allocated resources between two rooms). The parameters (see Fig. 4) for our Hill-Climbing algorithm are described as follows:

- *resource search*, the resource to be allocated can be selected at random or we may choose the worst offender of all, i.e. the resource whose removal causes the least penalty increase.
- *room search*, once the resource to be allocated has been selected, the room to be assigned can be selected at random, the best of a number X of rooms or the best in the list of available rooms.
- s*pace deviation*, is the percentage of space that can be wasted or overused when assigning resources to a certain room. This parameter depends on the university requirements.
- *termination criteria,* can be either a fixed number of iterations (modifications/attempts) or until there is no improvement in the solution after a certain number of attempts or modifications.

With these parameters, the *system administrator* can configure the two options that will be available from the *system user* level: a quick attempt of optimisation to produce acceptable solution quality, or a longer process to produce a higher quality solution. Typical values for a quick attempt are: random resource search, random room search, space deviation according to the specific requirements, termination criteria with a fixed number of iterations set to 100 times the number of resources. For a more thorough search process the common parameters are: random resource search, best of X rooms search (X set to one fifth of the number of rooms), space deviation according to the specific requirements and termination criteria with no improvement after a certain number of iterations set to 10 times the number of resources.

4 RESULTS AND EARLY CONCLUSIONS

This system has initially been tested with real data obtained from the University of Nottingham, this and other data collections can be found on the web [7]. In this case, the Computer Science Department occupies one building with 90 rooms. There are 117 resources classified as follows: 6 professors, 9 laboratories, 9 meeting rooms, 10 technicians, 5 storage rooms, 1 teaching assistant, 3 senior lecturers,

7 secretaries, 47 researchers, 19 lecturers and 1 visiting lecturer. There are 51 specific constraints. In the existing solution, all resources are allocated and all rooms used. Results obtained by the system are shown in table 1 compared with the values for the existing allocation.

Table 1 Results

Fitness Statistics	Automated Optimisation		Existing Solution
	Slow	Quick	
Resources Allocated	117	117	117
Rooms Used	90	90	90
Space Utilisation	82.45%	81.33%	77.99%
Resources Penalty	714.87	2689.86	1264.21
Space Wastage Penalty	479	499.4	639.8
Space Overuse Penalty	403.26	2867.01	17400.27
Total Penalty	1597.13	6056.27	19304.28
Time Taken (h:m:s)	0:29:53	0:05:22	--------
Iterations	20000	15000	--------

We can observe in Table 1 that our system improved the existing allocation in terms of space utilisation due to the fact that less space is being wasted or overused. Also, fewer specific constraints are violated yielding a lower penalty. The great reduction in the total penalty value is because of the substantial decrease in the space overused. Our conclusion is that our system offers estates managers a useful tool to automate the process of evaluation and if required, to improve the existing room distribution. The system has features that make it easily configurable by the administrator according to specific requirements; easy to use by managers; facilitate the visualisation of information related to the current process to evaluate the solutions proposed and reduce substantially the time necessary to solve space allocation problem. Future research work includes: hybridisation of our developed algorithms (Hill-Climbing, Simulated Annealing, Tabu Search, Genetic Algorithm) to solve the problem of constructing a complete new allocation, problem decomposition to tackle large space allocation problems and investigation on new approaches such as variable neighbourhood search and multi-criteria analysis.

REFERENCES

1. Aarts Emily, Lenstra Jan Karel, “Local Search in Combinatorial Optimisation”, Wiley, 1997.
2. C. Benjamin, I. Ehie, Y. Omurtag, “Planning Facilities at the University of Missoury-Rolla”, Journal of Interfaces, Vol. 22, No. 4, pp. 95-105, 1992.
3. Bland J.A., “Space-planning by ant colony optimisation”, International Journal of Computer Applications in Technology, Vol.12, No.6, pp.320-328, 1999.
4. E.K. Burke, D.B. Varley, ”Space Allocation: An Analysis of Higher Education Requirements”, The Practice and of Automated Timetabling II: Selected Papers from the 2nd International Conference on the Practice and Theory of Autometed Timetabling (PATAT’97), University of Toronto, 20-22 August 1997, Lecture Notes in Computer Science, Vol. 1408, pp. 20-33, Springer-Verlag 1998.
5. [Burke98a] E.K. Burke, D.B. Varley, “Automating Space Allocation in Higher Education”, Selected Papers from the 2nd Asia Pacific Conference on Simulated Evolution and Learning, Camberra, Australia, Spring Lectures Notes in artificial Intelligence, Vol, 1585, pp 66-73, 24-27 November 1998.
6. E.K. Burke, P. Cowling, J.D. Landa, ”The Space Allocation Problem”, http://www.asap.cs.nott.ac.uk/ASAP/space/spaceallocation.html
7. E.K. Burke, P. Cowling, J.D. Landa, ”Space Allocation Project – Test Data”, http://www.asap.cs.nott.ac.uk/ASAP/space/spacedata.html
8. Dominnie C.B., Kwak N.K., “A Hierarchical Goal Programming Approach to Reverse Resource Allocation in Institutions of Higher Learning”, Journal of the Operational Research Society, Vol 37, No 1, pp. 59-66, 1986.
9. Giannikos, E. El-Darzi, P. Lees, “An Integer Goal Programming Model to Allocate Offices to Staff in an Academic Institution”, Journal of the Operational Research Society, Vol. 46, No. 6, pp. 713-720, 1995.
10. Kim K.H., Kim H.B, “The optimal determination of the space requirement and the number of transfer cranes for import containers”, Computers & Industrial Engineering, Vol. 35, No.3-4, pp.427-430, 1998.
11. Larson N., Kusiak A., “Work-in-Process Space Allocation - A Model and an Industrial Application”, IIE Transactions, Vol. 27, No. 4, pp. 497-506, 1995.
12. Rayward-Smith, Osman, Reeves, Smith, “Modern Heuristic Search Methods”, Wiley, 1999.
13. L. Ritzman, J. Bradford, R. Jacobs, “A Multiple Objective Approach to Space Planning for Academic Facilities”, Journal of Management Science, Vol. 25, No. 9, pp. 895-906, 1980.
14. Zufryden F.S., “A Dynamic-Programming Approach for Product Selection and Supermarket Shelf-Space Allocation”, Journal of the Operational Research Society, Vol.37, No.4, pp. 413-422, 1986.