Office space allocation optimization

Conference Paper · May 2010

DOI: 10.1109/SIEDS.2010.5469670 · Source: IEEE Xplore

3 authors, including:

Rex K. Kincaid
College of William and Mary
71 PUBLICATIONS **676** CITATIONS

SEE PROFILE

Office Space Allocation Optimization

Rui Pereira*, Kevin Cummiskey** and Rex Kincaid***

***Abstract*— In large organizations, the allocation of buildings and office space to departments and employees is a challenging task. Optimal office space allocation has the potential to maximize synergies between employees within an organization. In this paper, we study the performance of a greedy search algorithm and a tabu search algorithm for generating high quality solutions to the office space allocation problem. The objectives are to maximize synergies in the organization, minimize the overusage of limited office space and maximize the number of buildings and rooms that can be completely closed. Computational experiments show that a tabu search algorithm generates higher quality solutions than a greedy local search algorithm with the same computational budget.**

I. INTRODUCTION

In large organizations, the allocation of buildings and office space to departments and employees is a challenging task. For example, in 2004, the NASA Langley Research Center reorganized and reallocated personnel. This move consisted of assigning workspace to more than 3,500 individuals and 1,600 research labs in approximately 6,200 rooms and 300 buildings. Reorganizations of this scale or larger occur frequently in the business sector as mergers and acquisitions create the need for a top-to-bottom review of an organization's footprint.

The purpose of this paper is to investigate the performance of a tabu search heuristic for finding high-quality solutions to office space allocation problems. Office space allocation problems consist of assigning employees workspace in an office building in an optimal fashion according to certain objective criteria. Employees can be distinguished by rank (Professor or Assistant Professor, for example) or by functionality (Engineer or Analyst, for example). In addition, employees are frequently given organizational assignments based on their membership in some subgroup of the organization. In practice, it is desirable to have several organizational assignments for each employee which indicates a more complex organization structure than a single-tiered structure.

There are two cases of the office space allocation problem that occur in practice. The first case is the complete reassignment of every employee in the organization to a new workspace. This occurs when an organization moves to a new building or undergoes some other transformation, such as a merger. In this case, it is economical to consider a full-scale reassignment of office space. The second case is the more frequent case where workspace is assigned to incoming personnel as the result of hiring and personnel turnover. In this case, strong consideration must be given to solutions that minimize the disruption to the workforce. Only the first case is considered in this study.

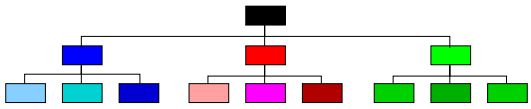

Fig. 1. Typical Organizational Structure

A. Literature Review

The office space allocation problem is a variant of the bin packing problem, the knapsack problem, and the generalized assignment problems [1,14], all known to be in the class of NP-complete problems. A related research problem is office space allocation in a university setting. A research group at the University of Nottingham (see references [1]-[5],[13]) has a series of papers on this topic. These focus on heuristic and meta-heuristic approaches to finding good solutions to the office space allocation problem. Solution approaches examined include genetic algorithms, simulated annealing, tabu search and asynchronous cooperative mechanisms [1]-[5],[13]. The population based genetic algorithms were outperformed by hill climbing and simulated annealing [10]. In [13], computational experiments indicated that population based variants of tabu search generated the best solutions from among local search and simulated annealing population based aysnchronous cooperation mechanisms.

Tabu search is a metaheuristic procedure that first appeared in the mid 1980's for solving optimization problems. It has been applied with much success to a wide variety of optimization problems including employee scheduling in [6], machine scheduling in [12], quadratic assignment problems in [15], and many others.

Currently, a NASA Langley Research research group employs a greedy local search algorithm for allocating employees from various organizations and divisions to research labs and offices [10]. Figure 2 and Figure 3 show NASA's initial and final office space allocations, respectively, where each block represents a building and each color represents a different suborganization.

This work was supported by the Old Dominion University Research Foundation.

*R. Pereira is a Graduate Student in Computer Science with specialization in Computational Operations Research at the College of William and Mary Williamsburg, VA 23185, USA rmpere@math.wm.edu

**K. Cummiskey is a Graduate Student in Computer Science with specialization in Computational Operations Research at the College of William and Mary Williamsburg, VA 23185, USA kfcummiskey@wm.edu

***R. Kincaid is a Professor in the Department of Mathematics at the College of William and Mary Williamsburg, VA 23185, USA rrkinc@math.wm.edu

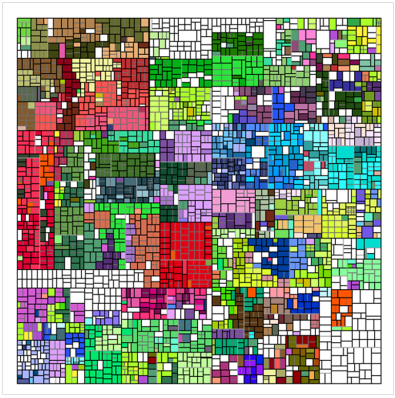

Fig. 2. NASA Initial Allocation

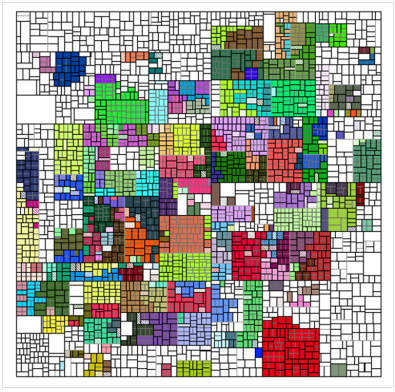

Fig. 3. NASA Final Allocation

II. PROBLEM STATMENT

Our objectives are to minimize the distance between employees in the same organization, minimize office space misallocation and maximize the number of vacant rooms.

$$g(x) = \alpha f_1(x) + \beta f_2(x) - \gamma f_3(x)$$

The objective function seeks to capture the following characteristics of candidate solutions: synergy, office space misallocation and building usage. We assume that the synergies between employees is greater when they are closer. In other words, the synergy between two employees is inversely proportional to their distance from each other. Applying this concept to the organization as a whole, solutions that minimize the sum of the distances between every pair of employees promote the greatest degree of cooperation and synergistic behavior. In order to reflect the importance of suborganization affiliation, distances between employees of the same suborganization are weighted. Therefore, the first term in our objective function minimizes the sum of the weighted distances between all employees.

The second term of the objective function is office space misallocation. Employees have square footage requirements based upon their position and job description. For example, lab technicians require a laboratory which typically occupies a larger amount of space. Similarly, management has specific space requirements. A penalty is added to the objective function if the size of the office assigned is less than the target size. The penalty is defined by a convex, piecewise linear function to generate an increasingly larger penalty for larger violations. The third term of the objective function counts the number of vacant rooms associated with a particular solution and adds this contribution to the objective function. This term reduces the value of the objective function because we wish to maximize the number of vacant rooms associated with a particular solution.

Figure 4 depicts two solutions that illustrate our objectives on a small example. Note that the better solution is the one in which five employees of the same organization are in the same room. Such a solution is feasible when the number of employees in an organization is divisible by the capacity of the rooms and all employees have identical square footage requirements. In addition, better solutions have more vacant rooms; here eight rooms are vacant after optimization.

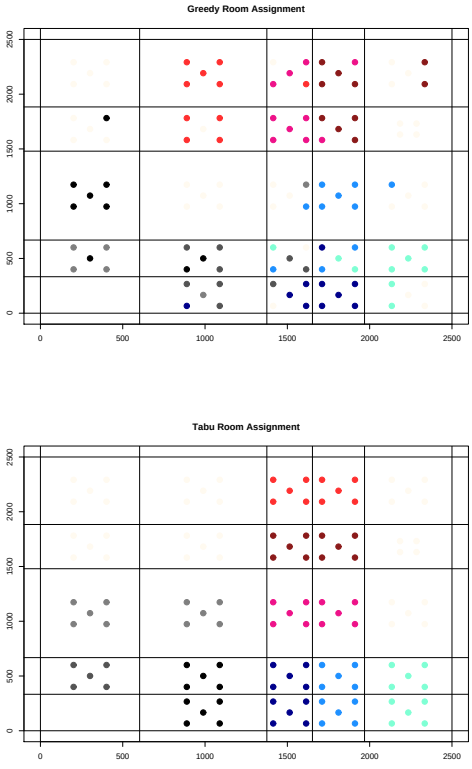

Fig. 4. Example Solutions

III. SOLUTION APPROACH

Our problem space consists of all possible allocations of `NumEmploy` employees to `NumRooms` rooms. Figure 5 depicts a small stylized representation of the problem. The simple problem has eight rooms. Our analysis focuses on placing employees in rooms, not allocating specific space to each employee in each room. Thus, we represent the centroid of each room by a node and the weight of the node reflects the number of employees present at that node. We

are interested in computing synergies between all employees in all rooms where each edge denotes a possible synergistic connection. Clearly, as we are dealing with a complete graph to model the synergies, the number of potential solutions increases significantly with each additional room and with each increase in the number of employees. As a result, total enumeration of all of the possible solutions is impractical even on relatively small real-world problems. In addition, we consider a two-tiered organizational structure with three larger organizations each with three divisions. This results in more possible solution combinations.

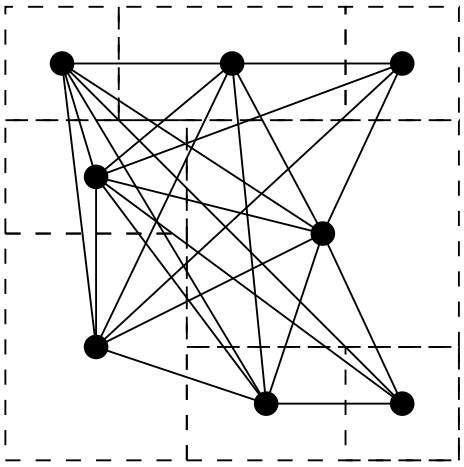

Fig. 5. Stylized Complete Graph

A. Solution Algorithms

Due to the complexity of the problem, we explore two heuristic approaches to generating high quality solutions, a greedy local search algorithm and a tabu search algorithm. Each of the algorithms is presented in pseudo-code below.

A greedy local search algorithm works by implementing discrete improvements from among a neighborhood of feasible solutions. A move, $\Delta x^{(t)}$, is defined to be a feasible change in the current solution vector. The set of all feasible moves, when applied to a given solution $x^{(t)}$, constitutes the neighborhood of candidate solutions for $x^{(t)}$. The algorithm proceeds through the candidate solutions in the move set, keeping track of, and ultimately updating, the current solution with a move that minimizes the objective function among the set of candidate solutions. The choice of an appropriate neighborhood influences the candidate solutions that are explored and therefore the quality of the best solution found over the course of the search.

We use the same move set for both the greedy local search and the tabu search algorithm. The move set includes two distinct types of moves. The first type of move swaps the room assignment for any two employees. The second type of move allocates an employee to an empty available position in one of the rooms. Initially, we also allowed a complete swap of all employees in different rooms, but later abandoned this type of swap for reasons explained in the next paragraph.

Greedy Local Search Algorithm

zBest $\leftarrow$ +inf
while(not improving)
  for $i = 1$ to N
    for $j = i$ to N
      swap employees i and j
      compute ΔObjective
      if ΔObjective < 0
        zBest $\leftarrow$ Objective
        store best solution
  Update $x^{(t+1)} \leftarrow x^{(t)} + \Delta x^{(t+1)}$

The complete room swap was found to generate improvements to the local solution with problem instances in which individuals, such as managers or lab technicians, require an entire room. The individual employee swaps are unlikely to locate and swap the workspace assignment for individuals with their own rooms due to the large number of potential swaps possible. Thus, the room swap allows for these individuals to have their room assignments change where they would otherwise not when restricting moves to individual employee swaps alone.

Tabu search can be applied to any process that involves moving from one candidate solution to another in a particular neighborhood where the quality of a candidate solution is determined by evaluating an objective function. Naive searches without a tabu search or other metaheuristic implemented easily become trapped in local optima. Tabu search is designed to escape local optima by avoiding reversals of moves that led to the local optimum.

Although there are many features that can be included in a tabu search, the two most common are a short-term memory list and an aspiration criteria. The purpose of the short-term memory is to prevent cycling by not allowing recently accepted moves to be repeated. Each time a move is accepted, it is added to the short-term memory list and remains 'tabu' for a user-defined number of iterations. The length of time a move attribute is tabu will influence the behavior of the search. The second feature commonly used in tabu searches is an aspiration criteria. Moves that meet the aspiration criteria override the short-term memory and allow the search to proceed with the tabu move. A common aspiration criteria is to allow moves that generate solutions better than the current best solution. A complete review of the tabu search heuristic can be found in [9] (see also, [7,8]).

Our implementation of the tabu search algorithm has two short-term memory lists. The first short-term memory list prevents swapping the same employees in subsequent iterations to prevent cycling. Due to the relatively large number of employees in the organizations, these moves remain in short-term memory for 150 iterations. A list length of 150 iterations was selected based on preliminary trials. The second short-term memory list prevents swapping employees in organizations that have recently been swapped. This second list is driven by the fact that the key element

driving the weights on the distance between employees is their organizational affiliations. As a result, individual employee assignments and organizational assignments can contribute equivalent weights to the objective function. Thus, the second memory list prevents swapping an employee from organization **B** with an employee from organization **A**, and then subsequently swapping a different employee from organization **B** with an employee from organization **A**, which would result in the same effective distance between employees of the same group.

Tabu Search Algorithm

zBest $\leftarrow$ +inf
for $i = 0$ to $IterNum$
  while(not improving)
    for $i = 1$ to N
      for $j = i$ to N
        swap employees i and j
        compute ΔObjective
        if ΔObjective < 0 and move is not tabu
          ΔBest $\leftarrow$ ΔObjective
          store best solution
        elseif ΔObjective $< \Delta$Best and move is not tabu
          ΔBest $\leftarrow$ ΔObjective
        elseif ΔObjective < 0 and move is tabu
                                and is the best move found
          ΔBest $\leftarrow$ ΔObjective
          store best solution
  Update $x^{(t+1)} \leftarrow x^{(t)} + \Delta x^{(t+1)}$

Both the greedy local search and the tabu search can update the solutions found based upon whether the solution is the best solution found from all possible swaps or, alternatively, after finding the first improving solution [15]. Computational experiments demonstrate that using as little as 30% of the possible moves in the relevant neighborhood can yield a substantial improvement in runtime for the algorithms while generating minor reductions in the quality of the best solution found[1]. This results from the fact that by searching over permutations of the solution vector we can obtain solutions that dominate a random constant ordering of the swaps the algorithm searches over [11,15]. As a result, we effectively search over 30% of the swaps, thus dramatically reducing the size of the problem. By searching over permutations of the solution vector we ensure that the tabu search algorithm can probabilistically search over many potentially good swaps, accepting the first-improving swap from among the candidate solutions. In addition, if the tabu search algorithm does not find any improving swaps for more than a certain number of iterations (this is a problem-size specific value), we enlarge the neighborhood size to find better quality solutions at the expense of a greater run time.

[1]Results available from author upon request

B. Test Problems

To generate random test problems, two important elements are required, generating rooms and generating the initial employee allocations. First, to simplify the problem setup for purposes of visualization, we designed the rooms and building structure around a grid. The solution algorithms use the room centroids and should therefore be relatively unaffected by this simplification. First, the buildings are generated and normalized by alternating the placement of vertical and horizontal walls at random locations with the defined area. Walls that result in any room with an aspect ratio of less than a third (a very tall and narrow room) or greater than three (a very short and wide room) are rejected and a new wall is placed. Next, `NumEmployees` are randomly placed in the building and assigned to rooms. The distance between any two employees is measured by the Euclidian distance between the centroids of the two rooms that they occupy. Employees that are located in the same room have zero distance between them.

Table I lists the test problems considered in this paper. These test cases are used to set up our problem implementation for real world applications. Smaller problems have a shorter run time and allow us to test the sensitivity of our solution algorithms and solution quality to key parameters.

We consider four sizes of test problems. The first test problem consists of 60 employees assigned to 12 rooms in a building. The second consists of 100 employees and 20 rooms; the third of 150 employees and 30 rooms. The final test problem consists of 210 employees in 42 rooms. Note that the rooms in this building are filled to capacity. Pretest simulations show that the runtime and solution quality depends on the density of the problem, that is the number of employees relative to available office space[2].

		Greedy	Tabu
1	Replications	1700	50
	Total Time	26.85	25.48
	NumEmployees	60	60
	NumRooms	12	12
	m^2	1200	1200
2	Replications	525	25
	Total Time	66.67	67.94
	NumEmployees	100	100
	NumRooms	20	20
	m^2	2000	2000
3	Replications	300	10
	Total Time	191.14	191.14
	NumEmployees	150	150
	NumRooms	30	30
	m^2	3000	3000
4	Replications	200	10
	Total Time	490.83	490.83
	NumEmployees	210	210
	NumRooms	42	42
	m^2	4200	4200

TABLE I
DESCRIPTION OF EXPERIMENTS

[2]Results available from author upon request

IV. RESULTS

Our results indicate that our tabu search procedure generates higher quality solutions than our greedy local search algorithm. The greedy local search procedure has a shorter run time than the tabu search algorithm. As a result, in order to level the playing field with respect to CPU time, we restart the greedy local search procedure from multiple random initial allocations in order to compare the solution quality with that of the tabu search algorithm. Table II presents the central results of our computational experiments. All computational experiments were performed on identical machines.

For the test problems with 100 employees, 450 iterations of the tabu search procedure took an average of 2.71 seconds. This is the equivalent runtime for 22 randomly restarted greedy local searches. The median iteration number in which the best solution was found for this problem is 318 iterations. The minimum tabu search solution over 50 replications, however, is 90.35% of the minimum solution found over 525 greedy solutions. Furthermore, the maximum solution observed for the tabu search algorithm is over the 50 tabu solutions is 15.14% greater than the minimum local search solution using equivalent CPU time.

For the test problems with 150 employees, 750 iterations of the tabu search procedure took an average of 19.11 seconds. This is the equivalent runtime for 31 randomly restarted greedy local searches. The median iteration number in which the best solution was found for this problem is 519 iterations. The minimum tabu search solution over 10 replications, however, is 96.72% of the minimum solution found over 300 greedy solutions. Furthermore, the maximum solution observed for the tabu search algorithm over the 10 tabu solutions is 7.3% greater than the minimum local search solution using equivalent CPU time.

For the test problems with 210 employees, 750 iterations of the tabu search procedure took an average of 49.08 seconds. This is the equivalent runtime for 20.2 randomly restarted greedy local searches. The median iteration number in which the best solution was found for this problem is 600 iterations. The minimum tabu search solution over 10 replications, however, is 98.79% of the minimum solution found over 200 greedy solutions. Furthermore, the maximum solution observed for the tabu search algorithm over the 10 tabu solutions is 10.51% greater than the minimum local search solution using equivalent CPU time.

Figure 6 depicts the solution quality for the computational experiments on the four test problems considered using the same y-axis scale. The tabu search procedure generates a statistically significant improvement over the greedy solution for the mean solution found. Due to the nature of the problem, however, we are typically more interested in the best case performance for equivalent CPU time. Again, we find that in all instances the tabu search algorithm outperformed the greedy local search algorithm. Figure 7 shows the individual comparisons for the solution quality for the four test problems. In each case it is clear that the tabu search generates improvements over the greedy local search using equivalent computational effort (note that the scale differs from axis to axis to illustrate more clearly the differences in solution quality between the greedy local search and the tabu search algorithms).

		Greedy	Tabu	Ratio
60	Mean	63833.8	45118.0	0.7068
	Min	46309.3	41840.8	0.9035
	Max	91146.0	49725.3	0.5456
100	Mean	298832.3	252777.2	0.8459
	Min	245811.8	233505.2	0.9499
	Max	397970.9	283021.8	0.7112
150	Mean	985840.7	884861.6	0.8976
	Min	865374.5	836996.4	0.9672
	Max	1146298.0	928910.2	0.8104
210	Mean	2695814.0	2541620	0.9428
	Min	2447139.0	2417588	0.9879
	Max	3001661.0	2704359	0.9010

TABLE II
RESULTS: OBJECTIVE VALUES

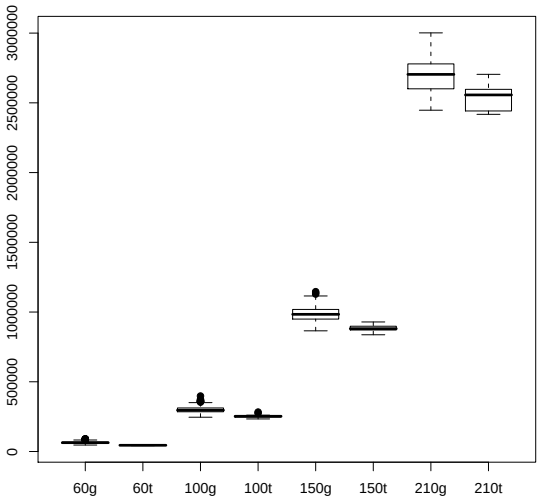

Fig. 6. Comparison of Solution Quality for the Greedy and Tabu Search Algorithms

Table III shows the results of the tabu search procedure, with and without the aspiration criteria. These experiments are conducted on the four test problems over 50 replications. In addition, note that in contrast with our central results these consider a more aggresive increase in the neighborhood size if the tabu search procedure does not find an improving solution within 20 iterations. Thus, the solutions here are marginally better than those reported in table II at the expense of greater computational effort. Note that the aspiration criteria indeed generates improvements to the solution, although the objective function values are close. The problem with 150 employees, however, shows a reversal of this result.

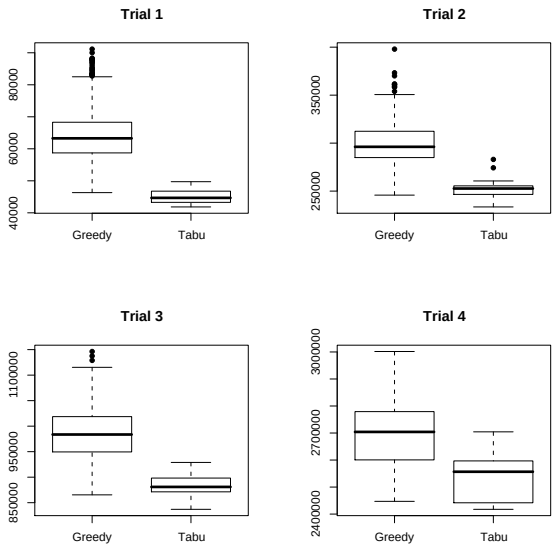

Fig. 7. Comparison of Solution Quality for the Greedy and Tabu Search Algorithms

		Aspiration Criteria		
		No	Yes	Number Asp
60	Min	41747	41446	18.1
	Mean	46231	45387	
	Max	51206	49338	
100	Min	233179	227913	28.5
	Mean	252304	250068	
	Max	281738	278221	
150	Min	832819	851995	44.8
	Mean	895406	904493	
	Max	972618	990061	
210	Min	2386435	2374625	59.2
	Mean	2532619	2571492	
	Max	2689244	2707518	

TABLE III
ASPIRATION CRITERIA

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

Tabu search is an effective technique for obtaining high quality solutions to office space assignment problems. For every problem size considered, tabu search outperformed the multi-start greedy solution significantly. The advantage gained by using tabu search over a multi-start greedy heuristic increases as the complexity of the test case increases. The greater the number of levels of organizations and types of employees, the greater the advantage of the tabu search.

B. Future Works

It is our goal to apply the algorithm to real-world allocation problems. An example of such a case is the relocation of the NASA Langley Research Center in which a greedy heuristic was used. Another area of interest is to further explore the relationship between the test case parameters and the relative performance of the search procedures. One such relationship to consider is that of the ratio of employees to number of rooms in the building (employee density) to the performance of the heuristics.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of the Old Dominion University Research Foundation and reviewers' comments.

REFERENCES

[1] E.K. Burke and D.B. Varley. "'Automating space allocation in higher education."' *In Proceedings of the 2nd Asia Pacific Conference on Simulated Evoluation and Learning*, pages 6673, Camberra, Australia, 1998.

[2] E.K. Burke, P. Cowling, and J.D. Landa- Silva. "'Hybrid population-based metaheuristics approaches for the space allocation problem."' *In Proceedings of the 2001 Congress on Evolutionary Computation*, pages 232239, 2001a.

[3] E.K. Burke, P. Cowling, J.D. Landa-Silva, and B. McMullan. "'Three methods to automate the space allocation process in UK universities."' *In The Practice and Theory of Automated Timetabling III: Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT 2001), Lecture Note in Computer Science*, pages 254273, Springer, New York, 2001b.

[4] E.K. Burke, P. Cowling, J.D. Landa-Silva, and S. Petrovic. "'Combining hybrid metaheuristcs and populations for the multiobjective optimisation of space allocation problems."' *In Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECC 2001)*, 2001c.

[5] E.K. Burke, C. Beyrouthy, J.D. Landa-Silva, B. McCollum, and P. McMullan. "'Spacemap applying meta-heuristics to real world space allocation problems in academic institutions."' *In Proceedings of the 2004 International Conference on the Practice and Theory of Automated Timetabling (PATAT 2004)*, pages 441 456, Pittsburgh USA, 2004.

[6] F. Glover and C. McMillan. 1986. The General Employee Scheduling Problem: An Integration of MS and AI. *COMP. OPER. RES.* Vol. 13, No. 5. pages 563-574.

[7] F. Glover, "Tabu Search, Part I," *ORSA J. on Computing,* **1**, pp. 190-206, 1989.

[8] F. Glover. 1990. Tabu Search: A Tutorial. *Interfaces*. Vol. 20, No. 4. pages 74-94.

[9] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.

[10] R. Kincaid, R. Gates, and R. Gage. "'Space allocation optimization at nasa langley research center."' *In Proceedings of the Seventh Metaheuristics International Conference*, Montreal, Canada, June 25-30, 2007.

[11] R. Kincaid and L.G. Yellin"The P-Dispersion-Sum Problem: Results on Trees and Graphs" *Location Science,* **1** (1993) pp. 171-186.

[12] M. Laguna, J.W. Barnes, and F.W. Glover. 1991. Tabu Search Methods for a Single Machine Scheduling Problem. *Journal of Intelligent Manufacturing*. Vol. 2, No. 2. pages 63-73.

[13] J.D. Landa-Silva, J. Dario, and Edmund K. Burke. Asynchronous cooperative local search for the office space allocation problem. *INFORMS Journal on Computing*, 19(4):575587, Fall 2007.

[14] S. Martello and P. Toth. *Knapsack Problems - Algorithms and Computer Implementations*. Wiley, New York, 1990.

[15] A. Ravi Ravindran (ed.) "Metaheuristics for Discrete Optimization," chapter (60 pages) in Operations Research and Management Science Handbook, , CRC Press, Taylor and Francis, December, 2007. ISBN: 9780849397219

[16] J. Skorin-Kapov. 1990. Tabu search applied to the quadratic assignment problem. *INFORMS Journal on Computing*. Vol 2. No. 2. pages 33.