PARETO OPTIMAL SOLUTIONS FOR MULTI-OBJECTIVE GENERALIZED ASSIGNMENT PROBLEM

S. Prakash[1], **M.K. Sharma**[2*] and **A. Singh**[3]

[1]Department of Applied Sciences
Amity School of Engineering & Technology, New Delhi, India
sprakash@aset.amity.edu

[2]School of Mathematics & Computer Applications
Thapar University, Patiala, India
mksharma@thapar.edu

[3]Department of Applied Sciences
Baba Banda Singh Bahadur Engineering College, Fatehgarh Sahib, India
amarinder77@gmail.com

ABSTRACT

The Multi-Objective Generalized Assignment Problem (MGAP) with two objectives, where one objective is linear and the other one is non-linear, has been considered, with the constraints that a job is assigned to only one worker – though he may be assigned more than one job, depending upon the time available to him. An algorithm is proposed to find the set of Pareto optimal solutions of the problem, determining assignments of jobs to workers with two objectives without setting priorities for them. The two objectives are to minimise the total cost of the assignment and to reduce the time taken to complete all the jobs.

OPSOMMING

'n Multi-doelwit veralgemeende toekenningsprobleem ("multi-objective generalised assignment problem – MGAP") met twee doelwitte, waar die een lineêr en die ander nie-lineêr is nie, word bestudeer, met die randvoorwaarde dat 'n taak slegs toegedeel word aan een werker – alhoewel meer as een taak aan hom toegedeel kan word sou die tyd beskikbaar wees. 'n Algoritme word voorgestel om die stel Pareto-optimale oplossings te vind wat die taaktoedelings aan werkers onderhewig aan die twee doelwitte doen sonder dat prioriteite toegeken word. Die twee doelwitte is om die totale koste van die opdrag te minimiseer en om die tyd te verminder om al die take te voltooi.

[*] Corresponding author.

1. INTRODUCTION

The generalized assignment problem, unlike other assignment problems, does not have a one-to-one correspondence between jobs and workers. Jobs are assigned in such a way that each job is assigned to a worker who may, however, be assigned more than one job, subject to the time available to him. The objective here is to minimise the total cost of assigned jobs. Many researchers, including Cattrysse & Wassenhove [2], Amini & Racer [1], Chu & Beasley [3], Diaz & Fernandez [4], Haddadi & Ouzia [8] and Kasana & Kumar [11] have proposed various methods – including exact and approximate ones – to solve the GAP and its variants. Lourenço & Serra [13] have applied heuristic techniques to solve the generalized assignment problem.

It is well known that multi-objective optimization problems have continued to attract a lot of attention due to their vast application. In multi-objective optimization problems, the objective functions are generally conflicting in nature. Thus the solution of the problem is important because of the trade-off relationship among the objectives. If the decision-maker knows the exact trade-off among the objective functions, then a single optimal solution will suffice; whereas, in the absence of explicit knowledge of the trade-off relationship among the objectives, it is better to find a set of Pareto optimal solutions first, and then choose one solution from the set, using the weights that may be set up by the decision-maker. Ignizio [9] has discussed multi-objective problems. Evans [5] has presented some reasons for the rapidly increasing interest in multi-objective mathematical programming, and has also discussed the advantages and disadvantages of a few general approaches to multi-objective mathematical programming. Lourenço et al. [12] have presented multi-objective metaheuristics for solving real-life crew scheduling problems in public transport companies. Fu & Diwekar [6] have presented an approach to multi-objective optimization based on the principles of probabilistic uncertainty analysis. Junker [10] has generalized a preference-based search to compute balanced, extreme, and Pareto optimal solutions for general constraint satisfaction problems, thus handling preferences on and between multiple criteria.

From a multi-objective point of view, Gandibleux & Freville [7] have used the Tabu search heuristic on the two linear objectives to solve the 0-1 knapsack problem. Zhang & Ong [14] have proposed an LP-based heuristic to solve the Bi-objective Generalized Assignment Problem (BiGAP), in which both objective functions are linear. In the present study, a BiGAP is considered in which both the objective functions are minimisation type performance measures. An algorithm has been proposed to solve this BiGAP without prioritising either of the objective functions, to obtain a set of Pareto optimal solutions. The Pareto optimal solutions are also termed ‘non-dominated solutions’ or ‘efficient solutions’.

2. FORMULATION OF THE PROBLEM

Consider the case of *m* workers and *n* jobs. Let c_{ij} *(i=1,2,…,m; j=1,2,…,n)* represent the cost of assigning the worker *i* to job *j*; t_{ij} *(i=1,2,…,m; j=1,2,…,n)* denote the time taken by the worker *i* to perform job *j*; x_{ij} *(i=1,2,…,m; j=1,2,…,n)* be the binary decision variable assuming the value 1 or 0 according to whether or not the worker *i* has been assigned the job *j;* and b_i *(i=1,2,…,m)* be the time available with worker *i*. Let *C* and *T* denote the total cost and duration respectively of assigning jobs. The mathematical formulation of the problem is as follows. Determine x_{ij}s, which minimise the two-objective functions:

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

$$T = \max \left\{ \sum_{j=1}^{n} t_{ij} x_{ij} \ :(i = 1,2,\ldots m) \right\} \tag{2}$$

without according priorities to them, subject to the constraints

$$\sum_{i=1}^{m} x_{ij} = 1 (j = 1,2,\dots n) \tag{3}$$

$$\sum_{j=1}^{n} t_{ij} x_{ij} \le b_i \ (i = 1,2,\dots,m) \tag{4}$$

$$x_{ij} = 0 \, or \, 1 (i = 1,2,\dots m; j = 1,2,\dots,n) \tag{5}$$

Here, (1) and (2) represent the objective functions that need to be minimised. Constraint (3) ensures that each job is assigned to just one worker. Constraint (4) ensures that the assignments to each worker are made only according to the time available to the worker. It is necessary to find the set of non-dominated solutions of the problem provided by Eqs. (1)-(5). A vector of decision variables $x^* \in \Omega$ for a k-objective problem is non-dominated if there is not another $x \in \Omega$ such that $f_i(x) \le f_i(x^*)$ for all $i = 1,2,\dots,k$ and $f_j(x) \le f_j(x^*)$ for at least one j.

3. SOLUTION PROCEDURE

The Bi-objective Generalized Assignment Problem is an integer nonlinear problem. This is so because the objective function provided by (2) is nonlinear, and the decision variables x_{ij}s assume the integer value 0 or 1. A procedure is outlined to obtain the set of non-dominated solutions.

3.1 Procedure to obtain first non-dominated solution

The first non-dominated solution is obtained by assigning the first priority to the objective function (1) and the second priority to the objective function (2). The algorithm is explained below:

Step 1. Calculate the least cost penalties for each job, i.e. the difference of least and second-least costs, for various workers.

Step 2. Select the job with the largest penalty. In the event of a tie in the largest penalty for two or more jobs, select the job for which the least cost of assignment is the lowest among all such least costs. In the event of a tie on that count as well, select the job that has the lowest time duration corresponding to the least cost cell; otherwise the job can be selected arbitrarily. Let the k^{th} job be selected.

Step 3. Select the least cost cell of the k^{th} job selected above. In the event of a tie in the least cost cells, select the one that has the lowest time duration in the least cost cells. In the event of a tie on the least time duration of the selected least cost cells as well, assign it to the worker who has the most time available; otherwise assign it arbitrarily. If the selected least cost cell *(r,k)* does not satisfy the time constraint, leave the job unassigned; otherwise make an assignment in the cell, i.e. assign job *k* to worker *r* and update the time available to him to b_r-t_{rk}. Drop the k^{th} job from further consideration.

Step 4. Repeat steps 2 and 3 until either each of the jobs has been assigned to a single worker, or a situation arises that some jobs are left unassigned because of the time constraints of the workers at their least costs. In the former case, a non-dominated solution is obtained; in the latter case, apply the next step.

Step 5. If some of the jobs are left unassigned, the job schedule of certain workers is subject to change – i.e. the allocation of combined jobs is considered for moving, preferring the least increase in cost, if an improvement in it is not possible, depending upon the following situations that may occur.

Step 5.1. An allocation of worker *r* is considered to move from job *j* to job *j′* if

$$\sum_{s=1}^{n} t_{rs} x_{rs} < b_r, (s \neq j)$$

where job *j′* is so chosen that

$$\sum_{s \neq j} c_{rs} x_{rs} < \sum_{s \neq j'} c_{rs} x_{rs}$$

if possible.

Step 5.2. A combination of assignments of worker *r* is considered to move from jobs *j* & *j′* to job *j** if

$$\sum_{s \neq j} c_{rs} x_{rs} < b_r$$

Once an assignment is shifted from the cell *(r,s)*, the cell is dropped from further consideration. Repeat the steps 5.1 and 5.2 until all the jobs have been assigned. The solution thus obtained will be termed the first non-dominated solution, denoted by $(C(X^{(1)}), T(X^{(1)}))$.

3.2 Procedure to obtain second and subsequent non-dominated solutions

After having obtained the first non-dominated solution by $(C(X^{(1)}), T(X^{(1)}))$, the second non-dominated solution $(C(X^{(2)}), T(X^{(2)}))$ is obtained by introducing an additional constraint

$$\sum_{j=1}^{n} t_{ij} x_{ij} < T(X^{(1)}) \, \forall i \tag{6}$$

to the formulation of the problem, to restrict the aspiration level of the decision-maker. The second non-dominated solution is obtained by dropping all the cells *(i,j)* in which $t_{ij} \geq T(X^{(1)})$ and assigning the jobs afresh by following the same procedure as explained in section 3.1, or by moving the assignments from the table of the first non-dominated solution in the following manner.

Step I. Select the workers who are working for $T(X^{(1)})$ amount of time, i.e., satisfy the following equality

$$\sum_{j=1}^{n} t_{ij} x_{ij} = T(X^{(1)})$$

Step II. Shift the assigned jobs of a worker to some other worker by assigning them at the available least cost cells, satisfying the constraints (3)-(6) as explained in steps 5.1 and 5.2.

Step III. Stop the procedure when all the jobs are assigned to workers, satisfying all the constraints (3)-(6).

The solution obtained is termed the second non-dominated solution denoted by $(C(X^{(2)}), T(X^{(2)}))$. The third non-dominated solution may be obtained by following the same procedure as for the second non-dominated solution just by replacing $T(X^{(1)})$ with $T(X^{(2)})$, and the same procedure can be extended to obtain subsequent solutions as well. The procedure terminates at the nth non-dominated solution $(C(X^{(n)}), T(X^{(n)}))$, when it is not possible to obtain a non-dominated solution $(C(X^{(n+1)}), T(X^{(n+1)}))$ where $T(X^{(n+1)}) < T(X^{(n)})$.

3.3 Some additional assignment procedures

More non-dominated solutions can be found by following the assignment procedures explained below.

Case I. Following the same penalties on cost, the assignments are made in the cells that have the least amount of time, satisfying the constraints (3)-(5). In the event of a tie in the cells for the least amount of time, the assignments are made in the cells that have a lower cost corresponding to the least time. In the event of a tie on that count as well, the assignment is made arbitrarily. If a solution is not obtained, i.e. not all the jobs are assigned, then the procedure is applied as explained in step 5 of algorithm in section 3.1. The solution should be compared with the non-dominated solutions already obtained for its dominance or non-dominance. The second and subsequent non-dominated solutions can be obtained by following the same procedure as explained earlier.

Case II. Calculate the least time penalties for all the jobs for various workers – i.e. the difference between least and second least time duration of all the jobs by various workers – and assign the jobs to the worker with least cost, following the same procedure as explained in section 3.1, where the penalties are calculated on the time duration and obtain a set of non-dominated solutions.

Case III. Using the same least time penalties for all the jobs, assign the jobs at the least time duration, satisfying the constraints (3)-(5). Ties are broken as explained in Case I above, thereby generating a set of non-dominated solutions.

It is worth noting here that some of the solutions obtained using the above cases may dominate or be dominated themselves by the solutions obtained by the procedure explained in sections 3.1 and 3.2.

4. NUMERICAL EXAMPLE

Consider the following example of multi-objective GAP with *m=4, n=7* in Table 1.The first entry in each cell represents c_{ij} and the second represents t_{ij}. In order to obtain the first non-dominated solution of the problem, the objective functions (1) and (2) are assigned the first and second priorities respectively. The penalties shown in Table 1 are the least cost penalties for all the jobs. Apply the procedure explained in section 3.1 and try to assign the jobs in decreasing order of penalties.

J / W	1	2	3	4	5	6	7	Available Time (b_i)
1	9 4	7 8	8 3	12 10	9 10	12 8	15 7	15
2	10 3	5 10	8 5	11 6	15 3	16 10	20 2	12
3	4 12	3 12	10 2	9 2	12 7	14 9	12 10	20
4	8 7	7 6	8 5	10 4	10 9	14 9	10 12	14
Penalty	4	2	0	1	1	2	2	

Table 1 – Numerical problem with 4 workers and 7 jobs

The largest penalty is 4 for job 1, and the corresponding least cost occurs for worker 3; and since he has sufficient time available to him, the job is allocated to him, and the time available to him is updated to 8 units. The next largest penalty is 2, which corresponds with three jobs: job 2, job 6, and job 7. Since job 2 has the lowest of the corresponding least costs, it is considered for allocation. The least cost cell for job 2 occurs for worker 3, but the job cannot be allocated to him since he does not have the required time to perform the job. So job 2 is left unassigned for now. Next, job 7 is considered for allocation, since the least cost for assigning job 7 is less than that of job 6. The least cost for job 7 occurs for worker 4, who is allocated this job since he has the required time to perform the job. Thus the time available to worker 4 is updated to 2 units. Thereafter job 6 is considered for allocation, and is allocated to worker 1 whose available time is updated to 7 units. The next largest penalty is 1 which corresponds with two jobs: job 4 and job 5. There is a tie in the least costs for job 4 and job 5, but preference is given to job 4 since the least cost cell – i.e. (3,4) for job 4 – has a lower time duration than that in the cell (1,5) for job 5. Thus job 4 is allocated to worker 3, who has already been allocated job 1, and his available time is updated to 6 units. Now job 5 cannot be assigned at the least cost – i.e. to worker 1 – since he does not have the required time available to him. Thus job 5 is also left unassigned. The last job under consideration is job 3, which has the same least cost for three workers: worker 1, worker 3, and worker 4. Job 3 is allocated to worker 1 since he has the least time duration as well. Thus, so far job 2 and job 5 have not been assigned to any worker. Disregarding the least cost cells for these two jobs, calculate the cost penalties by considering the next two least costs. It can be seen that the penalty for both the jobs is 2. Preference is given to job 2, since the least cost of assigning job 2 is less than that for job 5. Thus job 2 is allocated to worker 2, and his available time is updated to 2 units. Job 5 still remains unassigned, since no worker has the required time available to him to perform the job. It is easy to check that job 5 remains unassigned. The status of the assignments after applying steps 1-4 of the algorithm given in section 3.1 is given in Table 2.

J W	1	2	3	4	5	6	7	Working duration
1			(1)			(1)		11
2		(1)						10
3	(1)			(1)				14
4							(1)	12

Table 2 – Initial assignment of jobs

Since job 5 has not been assigned to any worker, select the least cost cell for the same, i.e. the cell (1,5) that corresponds to worker 1. Now change the allocation of worker 1 so as to include the cell (1,5) in his allocation. Since the total time available to the worker is 15 units, and he is already working for 11 units of time, some other allocation of worker 1 has to be shifted to the cell (1,5). The two possible options to move are (1,3) or (1,6). Preference is given to moving the assignment from (1,6) to (1,5), since that reduces the cost of assignment, even though the working duration of worker 1 increases to 15 units. This iteration leaves job 6 unassigned. Drop the cell (1,6) from further consideration of assignments.

Repeating step 5 of the algorithm of section 3.1, now on job 6, the available least cost cell is either (3,6) or (4,6). Since the least cost and the time duration are the same in both cells, the allocation is made in (3,6) since worker 3 has a greater amount of time available to him. Now the assignment of worker 3 may be changed in two ways, i.e. by moving the assignments of cell (3,1) or (3,4) to the cell (3,6). But shifting the assignment from (3,4) to (3,6) is not possible, since it violates the time constraint for worker 3. Shift the assignment from (3,1) to (3,6), leaving job 1 unassigned, and update the time available to worker 3 to 9 units. Thus, the second iteration of step 5 of the algorithm leaves the cell (3,1) unavailable for further consideration.

The third and subsequent iterations of step 5 of the algorithm are given below in Table 3.

Iteration No.	Job to be assigned	Variables x_{ij} at level 1	Job left unassigned
3	1	$x_{13}, x_{15}, x_{22}, x_{34}, x_{36}, x_{41}$	7
4	7	$x_{13}, x_{15}, x_{22}, x_{34}, x_{37}, x_{41}$	6
5	6	$x_{13}, x_{15}, x_{22}, x_{34}, x_{37}, x_{46}$	1
6	1	$x_{11}, x_{13}, x_{22}, x_{34}, x_{37}, x_{46}$	5
7	5	$x_{11}, x_{13}, x_{22}, x_{34}, x_{37}, x_{45}$	6
8	6	$x_{11}, x_{13}, x_{26}, x_{34}, x_{37}, x_{45}$	2
9	2	$x_{11}, x_{13}, x_{26}, x_{32}, x_{34}, x_{45}$	7
10	7	$x_{11}, x_{13}, x_{17}, x_{26}, x_{32}, x_{34}, x_{45}$	-

Table 3 – Iterations of Step 5

Since all the jobs have been assigned to individual workers, thus satisfying all the constraints, we have obtained the first non-dominated solution denoted by $(C(X^{(1)}), T(X^{(1)}))$. The corresponding assignments are given in Table 4, and the total cost and duration of the assignment are given in Table 5.

J W	1	2	3	4	5	6	7	Working Duration
1	(1)		(1)				(1)	14
2						(1)		10
3		(1)		(1)				14
4					(1)			9

Table 4 – Job assignment of first non-dominated solution

Variables at Level 1	$C(X^{(1)})$	$T(X^{(1)})$
$x_{11}, x_{13}, x_{17}, x_{26}, x_{32}, x_{34}, x_{45}$	70 units	14 units

Table 5 – First non-dominated solution

To obtain the second non-dominated solution denoted by $(C(X^{(2)}), T(X^{(2)}))$, the cells *(i,j)* in which $t_{ij} \geq 14$ are dropped from consideration. The solutions may be obtained by following two procedures explained in section 3.2. Starting from the assignment table of first non-dominated solution, i.e. Table 4, the assignment of jobs 1 and 3 is considered for moving from the existing worker to another worker who satisfies the constraints (3)-(6). The move results in the second non-dominated solution given in Table 6.

Continuing the same procedure, the third non-dominated solution is obtained, as shown below in Table 7.

Variables at Level 1	$C(X^{(2)})$	$T(X^{(2)})$
$x_{11}, x_{13}, x_{26}, x_{27}, x_{32}, x_{44}, x_{45}$	76 units	13 units

Table 6 – Second non-dominated solution

Variables at Level 1	$C(X^{(3)})$	$T(X^{(3)})$
$x_{11}, x_{13}, x_{26}, x_{27}, x_{34}, x_{35}, x_{42}$	81 units	12 units

Table 7 – Third non-dominated solution

The non-dominated solutions maybe obtained by allocating afresh in Table 1 and satisfying the constraints (3)-(6), following the same procedure explained in section 3.1 and applied above, to obtain the first non-dominated solution. The solution obtained is given below in Table 8.

Variables at Level 1	$C(X^{(4)})$	$T(X^{(4)})$
$x_{11},\ x_{17},\ x_{26},\ x_{33},\ x_{35},\ x_{42},\ x_{44}$	79 units	11 units

Table 8 – Fourth non-dominated solution

It is easy to check that the fourth non-dominated solution displayed in Table 8 dominates the third non-dominated solution given in Table 7. So the solution of Table 7 is rejected, and that of Table 8 is redesignated as the third non-dominated solution. The procedure ends here, since it is not possible to assign the jobs and obtain some other non-dominated solution. Thus the non-dominated solutions obtained so far are given in Table 9.

	Cost of assignment	Duration of assignment
First non-dominated solution	70 units	14 units
Second non-dominated solution	76 units	13 units
Third non-dominated solution	79 units	11 units

Table 9 – Set of non-dominated solutions

However, more solutions to the problem may be obtained by applying each of the three cases explained in section 3.3. There is, though, only one non-dominated solution, shown in Table 10, obtained by applying the procedure explained in Case I of section 3.3.

	Cost of assignment	Duration of assignment
First non-dominated solution	83 units	8 units

Table 10 – Set of non-dominated solutions obtained by applying Case I

Applying the procedure explained in Cases II and III of section 3.3, we obtain more solutions that may actually dominate, or be dominated by, the solutions obtained earlier. The non-dominated solutions obtained by applying Case II and III are given in Tables 11 and 12 respectively.

	Cost of assignment	Duration of assignment
First non-dominated solution	84 units	9 units
Second non-dominated solution	74 units	11 units
Third non-dominated solution	72 units	12 units
Fourth non-dominated solution	71 units	13 units
Fifth non-dominated solution	66 units	15 units

Table 11 – Set of non-dominated solutions obtained by applying Case II

	Cost of assignment	Duration of assignment
First non-dominated solution	83 units	8 units

Table 12 – Set of non-dominated solutions obtained by applying Case III

Moreover, it is easy to check that the solution in which $T(X^{(n)})<8$ is not possible, since $t_{i6}\geq 8$ for all i - i.e. job 6 cannot be assigned for durations less than 8 units. Thus the final set of non-dominated solutions, in decreasing value of the first objective function (1), is given below in Table 13.

	Cost of assignment	Duration of assignment
First non-dominated solution	83 units	8 units
Second non-dominated solution	74 units	11 units
Third non-dominated solution	72 units	12 units
Fourtt non-dominated solution	71 units	13 units
Fifth non-dominated solution	70 units	14 units
Sixth non-dominated solution	66 units	15 units

Table 13 – Final set of non-dominated solutions of the numerical problem

5. CONCLUSIONS

The main contribution of this work is the study of the bi-objective generalized assignment problem in which one objective function is linear and the other one is non-linear. Both objective functions – the linear one on the cost of assignment of jobs, and the non-linear one on the duration of completion of the jobs - are minimised. The solution procedure involved moving the assignments of workers to restrict the aspiration criteria of the decision-maker, so as to search the smaller neighbourhood for the possible set of non-dominated solutions that give greater flexibility to the decision-maker.

6. REFERENCES

[1] **Amini, M.M. & Racer, M.** 1995. A hybrid heuristic for the generalized assignment problem, *European Journal of Operational Research*, 87, pp 343-348.

[2] **Cattrysse, D.G. & Wassenhove, L.N.V.** 1992. A survey of algorithms for the generalized assignment problem, *European Journal of Operational Research*, 60, pp 260-272.

[3] **Chu, P.C. & Beasley, J.E.** 1997. A genetic algorithm for the generalized assignment problem, *Computers & Operations Research*, 24, pp 17-23.

[4] **Diaz, J.A. & Fernandez, E.** 2001. A tabu search heuristic for the generalized assignment problem, *European Journal of Operational Research*, 132, pp 22-38.

[5] **Evans, G.W.** 1984. An overview of techniques for solving multiobjective mathematical programs, *Management Science*, 30, pp 1268-1282.

[6] **Fu, Y. & Diwekar, U.M.** 2004. An efficient sampling approach to multiobjective optimization, *Annals of Operations Research*, 132, pp 109-134.

[7] **Gandibleux, X. & Freville, A.** 2000. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objective case, *Journal of Heuristics*, 6, pp 361-383.

[8] **Haddadi, S. & Ouzia, H.** 2004. Effective algorithm and heuristic for the generalized assignment problem, *European Journal of Operational Research*, 153, pp 184-190.

[9] **Ignizio, J.P.** 1982. *Linear programming in single and multi-objective systems*, Prentice-Hall, Englewood Cliffs, New Jersey.

[10] **Junker, U.** 2004. Preference based search and multi-criteria optimization, *Annals of Operations Research*, 130, pp 75-115.

[11] **Kasana, H.S. & Kumar, K.D.** 2004. *Introductory operations research. Theory and applications*, Springer-Verlag, Berlin.

[12] **Lourenço, H.R., Paixao, J.P. & Portugal, R.** 2001. Multiobjectives metaheuristics for the bus driver scheduling problem, *Transport Science*, 35, pp 331-343.

[13] **Lourenço, H.R. & Serra, D.** 1998. Adaptive approach heuristics for the generalized assignment problem, *preprint*.

[14] **Zhang, C.W. & Ong, H.L.** 2007. An efficient solution to biobjective generalized assignment problem, *Advances in Engineering Software*, 38, pp 50-58.