A Knowledge-Based Technique for Constraints Satisfaction in Manpower Allocation

Khaireel A. Mohamed[1], Amitava Datta[2]*, and Ryszard Kozera[2]

[1] Institut für Informatik, Albert-Ludwigs-Universität Freiburg
Georges-Kohler-Allee, 79110 Freiburg, Germany
khaireel@informatik.uni-freiburg.de
[2] School of Computer Science & Software Engineering
University of Western Australia
Perth, WA 6009, Australia
{datta, ryszard}@csse.uwa.edu.au

Abstract. We show that knowledge-based techniques are as effective as mathematical techniques when satisfying constraints for solving manpower allocation problems. These techniques can be used to fulfill the corresponding local and global constraints based on the dynamic programming algorithm. It uses tools borrowed from genetic and simulated annealing algorithms, and fuzzy logic methodologies. The schedules produced by our algorithm match the best schedules produced by human experts.

1 Introduction

Manpower allocation (MA) problems are combinatorial optimisation problems with constraints that must be satisfied to arrive at feasible solutions (schedules). Traditionally, daily schedules in work places were prepared by hand [2, 6]. This class of problem arises in the management of manpower for organizations that provide round-the-clock services – assigning employees to the numerous available designated deployment posts. Consequently, scheduling manpower resources such as nurses, physicians, maintenance crew, and security personnel is important, as good schedules run entire organizations efficiently. Essentially, this task requires expertise and experience. It is necessary to ensure that the resultant schedules optimally match the skills of the available resources to the various conditions and requisites of the deployment posts.

Stochastically, there are many ways to achieve an algorithm that produces good schedules. Among the most promising are the methodologies that construe the use of linear and non-linear programming [7], genetic [4], simulated annealing [10], and artificial intelligence (AI) [1, 8] algorithms. These approaches attempt to find the lowest cost solutions that optimize given cost functions with respect to the associated constraints. Most researches on scheduling algorithms have concentrated mainly on regular performance measures in which an objective function is defined in terms of a set of resources (manpower) and a set of tasks (deployment posts). All the methods and techniques

* This author's research is partially supported by Western Australian Virtual Environments Centre (IVEC) and Australian Partnership in Advanced Computing (APAC).

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2659, pp. 100–108, 2003.

portrayed in this paper contribute to coming up with realistic and feasible rosters for scheduling manpower to various available deployment posts. The rest of the paper is organized as follows.

In Section 2, we define the structuring of our constraints for interpreting MA problems using knowledge-based techniques. Optimal schedule determination on more global perspectives through Sauer's [12] *reactive* and *interactive* scheduling representations is discussed in Section 3. In Section 4, we discuss the rule-based methodologies for backtracking through fuzzy logic techniques. In Sections 5, we discuss our results.

2 Problem Formulation

Brachman's [1] description of AI suggests two ways in which knowledge-based techniques can be implemented to solve the constraints in MA problems. Firstly, each resource's abilities and the requisites of the various posts for allocation must form the knowledge representation that can be reasoned to reflect human intelligence. Secondly, it suggests that the only way such techniques can behave in this manner is if they contain formal mechanisms for representing knowledge and employ inference techniques that model computationally based structures.

We can define the MA problem as follows. There is a given set of manpower resources R, each of whom has zero or more skills, and is identified by their distinctive ranks of seniority. Let u denote a set of skills, and v a set of ranks. Let $R[u \cup v]$ represent the set of resources with skills u and rank v. The set R is made up of n categories representing n different types of employee management. For example, the resources employed by a particular company are either in the "full-time", "part-time", or "temporary" staff category.

There is also a given set of deployment posts P; each of which has one distinct requirement, and a range of possible ranks for accommodation. Let s denote a set of requirements, such that $P[s \cup v]$ represents the set of deployment posts with a requirement s, and an accommodation of ranks v. Some of these deployment posts can be further grouped by the descriptive nature of their job specifications. Let Q be a set of similarly grouped deployment posts, so that $Q \subseteq P$. It is noted here, that there also exists a set of optional deployment posts in P that do not require to be filled immediately.

The resources to be scheduled must match their skills against the requirements of the deployment posts, and their ranks must be within the range that can be accommodated by those posts. This *skill-rank* composition constraint is imposed so that we can make direct constraint matchings between the resources and the deployment posts. Also, further global constraint rules are necessary, and are described below.

- *Deployment Post Prioritisation Constraints.* Mandatory deployment posts must take precedence over optional ones.
- *Officer-In-Charge (OIC) Constraints.* There must be at least one OIC for any groups of similar posts in Q.
- *Categorical Constraints.* Each set of similar posts Q should have more than one category amongst the resources assigned

MA problems may be formulated as a grid network [6], where each node represents a slot in the roster. The deployment posts' references form the horizontal constraints, while the resources' references form the vertical constraints. This method relates closely to the standard Constraint Satisfaction Problem (CSP). Unfortunately, it has been proven that CSP is *NP*-complete, even though the underlying network is a grid.

The output is a feasible solution with all resources assigned to the various deployment posts for any given workday. An assignment is deemed *feasible* if it satisfies the *skill-rank* composition constraints without leaving out any resources. On top of that, the assignment is considered *practical* if it also satisfies the three global roster constraints defined in the previous section.

2.1 Knowledge Representations and Relations

We can develop sufficiently precise base-notations for representing knowledge through the use of *logical representation schemes*, which employ semantics to describe a world in terms of objects and their binary associations [13]. These association types relate objects (resources and posts) to their generic types (skills and ranks), and force a distinction between the objects and the generic types.

Inferences and deductions can be subsequently obtained through direct matching processes. Matching as a knowledge-based operation is used for classification, confirmation, decomposition, and correction. The utilization of a uniform representation of knowledge in this technique enables the construction of tools that can verify desired properties of the knowledge representation [9].

Each element in the sets $R[u \cup v]$ and $P[s \cup v]$ is represented as a 32-bit binary strain. In this paper, we assume the number of ranks and skills, related to the resources and the posts, to be fewer than 15. However, our method is not restricted to this assumption and can be applied to an arbitrary number of ranks and skills. The lowest 15 bits of this binary strain represent the various ranks a particular resource can have, while the highest 15 bits represent the skills. This exact configuration is used to represent the ranks that can be accommodated by the deployment posts, as well as the requirements needed to make up the job specifications. In addition to this pattern, one of the middle two bits is reserved to note a preferred flag for calculating a preferred probability value. This bit is set to follow the reference knowledge given by human experts, should the resource or the post referred to by the strain has an additional preference in the *categorical* requirements.

We can assume that u is directly related to s, since the skills of a resource can be matched up against the requirement of a deployment post. A relations-matrix W is referred to throughout this paper as a 2D matrix; with references to the deployment posts forming the rows, and references to the manpower resources forming the columns. The construction of W is detailed in the next section.

We start by creating feasible solutions with pre-assigned, randomly generated resources for a set of deployment. These are then shuffled and fed as input to a database to test the implementation of our prototype. These random data resources are controlled by variables that include the total number of resources in relation to each rank, the distribution of ranks, and the distribution of skills. The information for the deployment

posts is also random. This ensures that the algorithm being tested is able to work on any set of data R, for any set of information P. We then set the final objective for the entire process, to minimize the number of un-assigned resources, while satisfying all local and global constraints.

2.2 Applying the Knowledge-Based MA Algorithm

Lau and Lua [6] gave a high level description of their rostering algorithm, on which they based their approach by using constraint set variables. Their algorithm has the following steps: **Step 1.** Perform feasibility checks; **Step 2.** Consider rostering constraints; **Step 3.** Generate a solution; **Step 4.** Apply local improvements.

Step 1 to 3 describes the Dynamic Programming (DP) algorithm, while *Step 4* is a backtracking rule-based mechanism that allows for additional improvements to be made to a feasible solution. We used the DP algorithm in the following context.

Firstly, we retrieve the information defined in R and P from the databases, and initialize their binary strains. Following which, we use the DP algorithm to perform the feasibility checks, solve the rostering constraints, and generate a feasible solution. This solution is checked to see if practical satisfaction is achieved *(see 2. Problem Formulation)*. Failing this, *Step 4* is executed to make local improvements by un-allocating resources that are perceived to have non-practical assignments. Lau and Lua termed this as the "relax & enrich" strategy of Schmidt in [6]. This is done through the use of additional constraints for an additional set of preferences to be met separately to avoid possible conflicts with the present ones. These constraints are part of the local improvement procedures. The process is then repeated until the number of unassigned resources reduces to zero.

3 Implementation and Methodologies

The DP algorithm above begins by building up the matrix W and shuffling the references to the resources and the posts for controlled fairness.

We used a knowledge-based technique to perform a local search for perfect matches between the row and column references of W. Each cell in W, referred to by a particular row and column, corresponds to a particular deployment post and resource respectively. A perfect match is found if the rank and skills possessed by a resource, matches that of the rank accommodations and the skill requirement of the deployment post as defined in Equation (1). The bit patterns defined in Figure 1 are used to contrast this matching process. A value of 1 is awarded to $W(i,j)$ should a perfect match be found, and 0 otherwise; here i and j are the row and column references to the relations matrix W.

$$W(i,j) = \begin{cases} 1, \text{ if } \{(u \cap s) \cup v\} > 0 \text{ where } s,u,v \in R,P \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

The result of the perfect matches allows the building of a more global relation between the resources and the posts. The DP rule for this relation is defined by Equation (2), where i and j are the row and column references to the relations matrix W. The

number in each cell of W describes the degree of relation between its rows and columns. The higher this number is, the better the relation.

$$W(i,j) = max\{max(\text{on row } i+1), max(\text{on column } j+1)\} + 1 \tag{2}$$

Probability functions are used to overcome the situation when more than one solution becomes inevitable. Our probability function resolves the outcome of a decision making process through a set of pre-determined rules. It ensures optimality of the local results with respect to the global outcome. We split this decision making into two separate components contributing to the final probability: the *measure of suitability*, and the *preferred probability*. The measure of suitability refers physically as to whether or not a particular resource is suitable to handle a particular post. The preferred probability, on the other hand, tries to model the human perception of *preferred*-ness in allocating the best resource to a particular post.

To calculate the measure of suitability of a candidate for a deployment post, we can build a reference probability vector from the deployment post information through its nature of binary strains arrangement. A value of 0.45 is allotted at every occurrence of a positive value in the strain (*1* being positive, and *0* being negative). We then take the dot product of this reference vector with a candidate from the column element. The result of which will be the measure of suitability for the candidate to the deployment post.

In calculating the preferred probability, the value of the *preferred* bit in the strain is checked. If the *preferred* bit in the deployment post's strain information matches the category type of a particular candidate, a full 0.50 is awarded. Otherwise, a random number between 0.00 and 0.25 is given to the preferred probability value.

Starting at $W(1,1)$, the entire first row and column are searched for cells with the highest degree of relation. These identified cells are then used as guides in obtaining the best probability rate, before allocating a selected column j (the most suitable resource candidate from the identified range) to a selected row i, (deployment post) as a complete assignment. Following this, that entire row i and column j are crossed out from any further comparisons.

The next iteration starts on the immediate row and column neighbour, $W(i+1, j+1)$, and ends when either the row or column elements of W are exhausted. The whole process is then repeated; rebuilding a new relations matrix W, after removing the entries of all successfully allocated resources and posts.

All the simulations in the experiment ended with all candidate resources allocated a post. It was also noted that it took fewer than 20 iterations to generate a feasible solution for a deployment problem with 12 distribution of ranks and 340 manpower resources. However, the scope of this algorithm at this point is limited only to solving the local constraints of the MA problem. These feasible solutions have yet to go through *Step 4* of the MA algorithm for completeness.

4 Rule-Based Methodologies for Backtracking

A more global approach is imperative in coming up with a better overall solution than above. Sauer [12] classified the task of scheduling to be either *reactive* or *interactive*. In reactive scheduling, all events occurring in the scheduling environment have to be

regarded, and then adapting the schedule to new situations with appropriate actions to handle each of the events. Interactive scheduling, on the other hand, means that several decisions about the constraints have to be taken by the expert human-scheduler. In the context of the MA problems, combining the two classes results in more effective and practical solutions. The knowledge-based techniques paired with the appropriate rule-based methodologies define the reactive and interactive representation to solving the local and global constraints respectively.

Gudes *et al.* [5] highlights the backtracking mechanism as a strategy to realize a new set of recommending rules. These rules suggest the de-allocation of one resource, and the allocation of another. The context here requires a different set of rule-lists to trigger the de-allocation procedure. In most backtracking scenarios, the completed roster is examined to identify assignments that are not "humanly" practical, that need to be taken off and reallocated elsewhere. For example, there can be too many officers-in-charge (OIC) for a group of similar deployment posts. The collective viewpoint to effect this de-assignment is to inspect the roster segregated into its associated groups, by the descriptive nature of the job specifications.

Zadeh's [14] conceptualization of fuzzy logic gives rule-based methodologies the ability to infer from the information resident in the knowledge-base. A membership function over a lattice of propositions can be formulated to encompass a wide range of logics of uncertainty. These logics make up the global rules for backtracking in our MA algorithm. These are in anticipation of the following *general rule of logic*:

If <*Number of resources*> AND <*Rank*> Then <*De-Allocate resources*>

Applying this general rule of logic requires the fuzzy logic concept of using *membership functions* with various associated degrees-of-freedom (DOF). A membership function is a graphical representation of the magnitude of participation of each input reference. It associates a weighting with each of the inputs that are to be processed, define functional overlaps between these inputs, and ultimately determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into rationalized output values that determine the decision-making.

A triangular shaped membership function is the most common one used in many control systems, mainly because of its mathematical simplicity. Plugging in prescribed input parameters from the horizontal axis, and projecting vertically gives the rationalized value of its associated DOF. By computing the logical product of the membership weights for each active rule, a set of fuzzy output response magnitudes are produced. All that remains is to combine and defuzzify these output responses, as an interpretation to our implementation of the backtracking mechanism.

Going by human semantics, and preparing for the *general rule of logic*, we can tabulate the input linguistic rules to give the appropriate output action rule. These output rules are suggested by the experts, and are shown in Table 1. For example, if there are *several* resources with *high ranks*, then we need to *de-allocate several* of the already assigned resources.

Table 1. Rule-base relations between the input linguistic parameters and the output actions.

Resources	Low Rank	High Rank
Too Few	None	Few
Several	None	Several
Too Many	Few	A lot

4.1 Fuzzification and De-fuzzification

The Fuzzification procedure converts the input parameters to a related fuzzy value with the associated DOF. We then used Mamdani's Maximum-Minimum Centre of Area method to calculate the output equivalence with respect to the rule-base given in Table 1 [8]. This de-fuzzification results in the actual number of resources to be removed with an attached DOF.

Our global backtracking rules look through each individual group of the segregated deployment posts with their assignments. The backtracking procedure parses information to pick out the number of resources and their associated ranks. The rule-base is then inferred, through fuzzy logic techniques described in Section 4, to determine the number of resources to de-allocate, and to await future assignments. This algorithm will continue to allocate and de-allocate resources to and from their posts in a generalized swapping manner, due to its absolute random nature of selection. The iteration will stop only when all resources are completely assigned.

The combined knowledge- and rule-based methodologies were tested with numerous sample data that are typical in representing the real-world situation. Randomised data for resources and posts were also generated to test the firmness of the algorithm.

Figure 1 depicts a typical result at each iteration step of the MA algorithm. We observe the distinctive sections in the figure where the knowledge-base techniques successfully reduced the number of un-allocated resources. At stipulated intervals, the rule-base methodologies kicked-in to de-allocate the resources that were deemed impractical for the roster. In fewer than 30 iterations, the solution generated by the MA algorithm is converted from feasible to a practical one.

5 Implementation and Experimental Results

We tested the robustness of the combined MA algorithm with the new addition of the dynamic rules through various numbers of randomly generated resources and posts. Figure 2 illustrates a histogram of results obtained from simulating 209 resources and posts. The histogram depicts the resultant data obtained by running 1023 different samples.

The histogram also indicates that a large proportion of the samples took less than 60 iterations before arriving at feasible and practical solutions. Depending on the tightness-level of correlation between the random resources and the random posts, the behaviour of the algorithm displays variations in its number of iterations when generating a roster. MA-type problems are *NP*-complete in nature [6], and there exists multiple solutions for

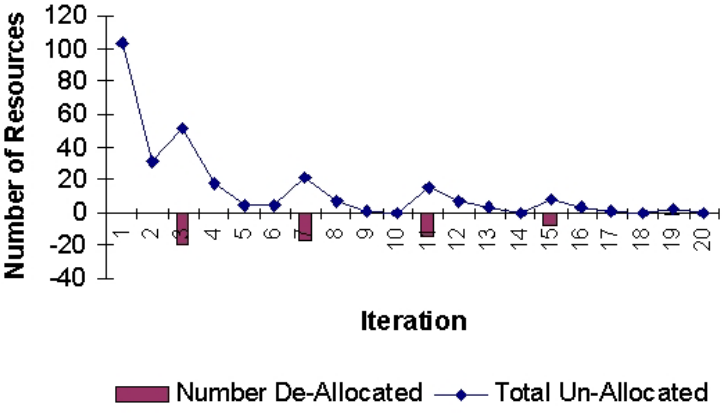

Fig. 1. Behaviour of the MA algorithm on the total un-allocated resources against the number of resources removed at each iteration. Data: Number of Posts = 151, Number of Resources = 103, Number of Ranks = 8.

any sample space that are optimal. Hence we can assume that the first optimal solution achieved is the accepted solution to be analyzed.

Mutalik *et al.* [10] and Ghoshray and Yen [3] used mathematical techniques in their approaches to solving constraints of optimisation problems. On the other hand, our knowledge- and rule-base improvement techniques use processor time for picking out identical strains, calculating probability functions (this is done only when more than one solution becomes inevitable), and resolving inference rules. All of the strains were preloaded into memory at initialization, thus making the comparison process simple but efficient.While the mathematical techniques in commercial scheduling softwares [13, 11] deal with non-integers to calculate the constraint function definitions, the error acquired at each function grows if the results obtained from one expression are reused again for another.Contrary to this, our knowledge-based techniques compare simple binary strains, and only deal with a one-off probability function for their decision-making. Clearly, the error rate here is not an issue.

References

1. R. J. Brachman. The basics of knowledge representation and reasoning. *AT&T Technical Journal*, 67(1):15, 1988.
2. Soumitra Dutta. Approximate spatial reasoning. In *Proceedings of the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 126–140. ACM Press, 1988.
3. S. Ghoshray and K. K. Yen. More efficient genetic algorithm for solving optimization problems. In *Systems, Man and Cybernetics. IEEE International Conference on Intelligent Systems for the 21st Century*, volume 5, pages 4515–4520. IEEE, Oct 1995.
4. Yu Gu. Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering*, 125(5):384–389, Sep 1999.

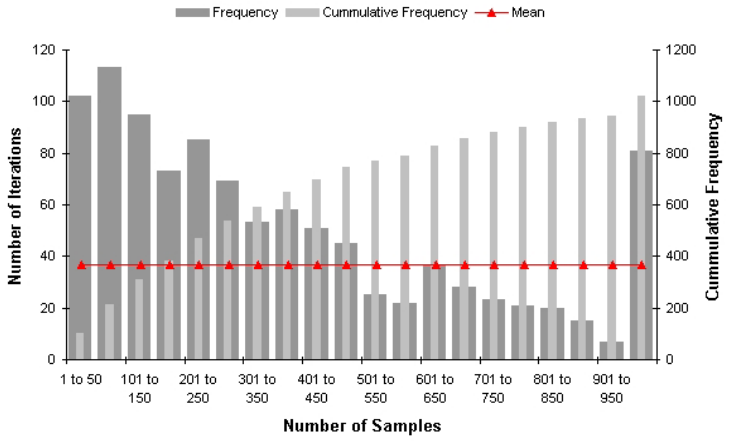

Fig. 2. Data obtained from simulating 1023 samples for 209 resources, 209 posts; 8 ranks, and 10 skills. The average number of iterations calculated is 367.152, with a standard deviation of 293.351. At $\alpha = 0.05$, the confidence interval is 17.977.

5. Ehud Gudes, Tsvi Kuflik, and Amnon Meisels. An expert systems based methodology for solving resource allocation problems. In *Proceedings of the Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 309–317. ACM Press, 1990.
6. H. C. Lau and S. C. Lua. Efficient multi-skill crew rostering via constrained sets. In *Proceedings of the Second ILOG Solver and Scheduler Users Conference*, July 1997.
7. David G. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley Publishing Company, 2nd edition, 1989.
8. E. H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. In *Proceedings of the Sixth International Symposium on Multiple-valued Logic*, pages 196–202, 1976.
9. L. J. Morell. Use of metaknowledge in the verification of knowledge-based systems. In *Proceedings of the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 847–857. ACM Press, 1988.
10. Pooja P. Mutalik, Leslie R. Knight, Joe L. Blanton, and Roger L. Wainwright. Solving combinatorial optimization problems using parallel simulated annealing and parallel genetic algorithms. In *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, pages 1031–1038. ACM Press, 1992.
11. Elmira Popova and David Morton. Adaptive stochastic manpower scheduling. In *Proceedings of the 30th Conference on Winter Simulation*, pages 661–668. IEEE Computer Society Press, 1998.
12. Jürgen Sauer. Knowledge-based scheduling techniques in industry. *Intelligent Techniques in Industry*, 1998.
13. Michel Turcotte, Gillian M. Mann, and Aaron L. Nsakanda. Impact of connection bank redesign on airport gate assignment. In *Proceedings of the 31st Conference on Winter Simulation*, pages 1378–1382. ACM Press, 1999.
14. Lotfi A. Zadeh. Coping with the imprecision of the real world. *Communications of the ACM*, 27(4):304–311, 1984.