COMPARISON OF METAHEURISTICS FOR WORKFORCE DISTRIBUTION IN MULTI-SKILL CALL CENTRES

David Millán-Ruiz
Telefonica Research & Development, Madrid, Spain

J. Ignacio Hidalgo
Complutense U. of Madrid, Madrid, Spain

Josefa Díaz
University of Extremadura, Mérida, Spain

Keywords: Memetic algorithms, Variable neighbourhood search, Iterated local search, Simulated annealing.

Abstract: Call centre technology requires the assignment of a large volume of incoming calls to agents with the required skills to process them. In order to determine the right assignment among incoming calls and agents for a real production environment, a comparative study of meta-heuristics has been carried out. The aim of this study is to implement and empirically compare various representative meta-heuristics, which represent distinct search strategies to reach accurate, feasible solutions, for two different instances of the workforce distribution problem. This study points out how memetic algorithms can outperform other acknowledged meta-heuristics for two different problem instances from a real multi-skill call centre from one of the world's largest telecommunications companies.

1 INTRODUCTION

A Multi-Skill Call Centre (MSCC) is a centralised office where there are groups of agents who may have a variable number of skills to handle large volumes of heterogeneous incoming telephone calls. Even though MSCCs have been widely studied, there are still some lacks on optimisation which may imply huge losses of money every year and client dissatisfaction due to everlasting delays.

A key feature of an MSCC is the Automatic Call Distributor (ACD). The ACD is a system which models incoming calls and automatically distributes them over different queues from which certain agents can pull work. The routing strategy is a rule-based set of operations that guides the ACD to process a given incoming call within the system. Typically, once the call has been classified and assigned to a Call Group (CG) in relation to its nature; a second algorithm either selects the best available agent to reply to a given incoming call or reconfigures the assignments *call group-agent*.

The basic variant of the workforce distribution problem in MSCCs requires the assignment of incoming calls to the agents who have the required skills to handle them over time, satisfying a given set of additional constraints and respecting dependencies among individual incoming calls and differences in the execution skills of the agents. In our case, these constraints are associated to incoming calls, agents, timings and desired/undesired actions.

Workload distribution in MSCCs has been generally faced by a Skill-Based Routing algorithm (SBR). Garnett (2000) defines SBR as a call-assignment protocol used in CCs to assign an incoming (customer) call to the most appropriate agent, instead of merely opting for the next existing agent. The major handicap of this approach is that online (ad-hoc) routing heuristics cannot be very complex in view of the fact that a very short response time is required. These fast, unplanned decisions may imply suboptimal task assignments to existing agents.

Nevertheless, Millán-Ruiz (2010) has recently demonstrated that Memetic Algorithms (MAs) can

Millán-Ruiz D., Hidalgo J. and Díaz J..
COMPARISON OF METAHEURISTICS FOR WORKFORCE DISTRIBUTION IN MULTI-SKILL CALL CENTRES.
DOI: 10.5220/0003083503520357
In *Proceedings of the International Conference on Evolutionary Computation* (ICEC-2010), pages 352-357
ISBN: 978-989-8425-31-7

outperform other classical call centre techniques (including SBR) when combining middle-term predictions with optimisation. But, given this formulation of the problem, other AI-style techniques could be also applied to this environment. The present paper investigates whether MAs can also outperform, for two different real-world instances of the problem of workforce distribution in MSCCs, other Meta-Heuristics (MHs) such as Iterated Local Search (ILS), Simulated Annealing (SA) or Variable Neighbourhood Search (VNS). These acknowledged techniques have been carefully chosen because they provide different search strategies to obtain feasible solutions.

The rest of this document is organised as follows: Section 2 describes the problem definition. Section 3 presents the encoding and the target function for the workforce distribution problem. Section 4 describes the MHs to be compared and their tuning. Section 5 conducts an analysis of the results. Finally, Section 6 concludes our work with a summary of major contributions and points out prospects for future work.

2 PROBLEM DESCRIPTION

In an MSCC, there are n customer calls queued in k CGs and m agents that may have up to l different skills s_i ($l \leq k$) to process these types of calls. Note that s_i represents the skill to process incoming calls from CG_i (s_i~CG_i). In an MSCC, each agent can process calls from different CGs and, given a CG, it may be answered by several agents who have the required skill. Obviously, this scenario can be simpler in some special CCs with only one CG in which agents have a single skill. These CCs can be modelled with q single queues working in parallel (all of them for the same CG). In other cases, every agent has all possible skills; hence all customer calls are queued in a single queue so that every agent can take and process any customer call. The system is noticeably easier to analyse in these two extreme cases. With all agents having all skills, the system is also more efficient (shorter waiting times, fewer abandonment rates) when the service time distribution for a given call type does not depend on the agent's skill set. However, this assumption turns out to be wrong in practice: agents are usually faster when they handle a smaller set of call types (even if their training gives them more skills). Agents with more skills are also more high-priced as their salaries depend on their skill sets. Thus, for large volumes of call types, it makes sense to dedicate various single-skill agents (specialists) to handle most of the load. A small number of agents, proportional to the calls of each type, with two or more skills can cover potential fluctuations in the arriving load.

To address the abovementioned fluctuations, the skills are grouped in skill profiles P_i. A skill profile P_i can be any subset of agent skills, containing up to l skills ($P_i=\{s_j(1), s_j(2),...,s_j(l)\}$) so that we can assign an agent to specific types of calls during a given period of time, despite this agent may have additional skills to process other type of calls. Figure 1 illustrates the relationship among clients' calls, queues and agents and Figure 2 shows an example of a feasible assignment. Millan-Ruiz (2010) provides further details of the problem of workforce distribution within MSCCs.

The solution to the problem of the workforce distribution in MSCCs is defined as the right assignment for every agent a_i to the most suitable skill profile P_j from his/her real skill profiles for each v seconds, where v is the size of the time-frame considered (in the CC studied, this must be done each *300* seconds). To determine whether (or not) a given solution is suitable, we need to define a quality metric to evaluate the rightness of each feasible solution. There are very significant metrics to measure the quality of a CC such as the abandonment and service rates. These metrics somehow hinge on the (customer) *service level* (Koole, 2006) which is defined as the percentage of customer calls that have to queue shorter than a specified amount of time. Our work has been conducted by applying this metric.

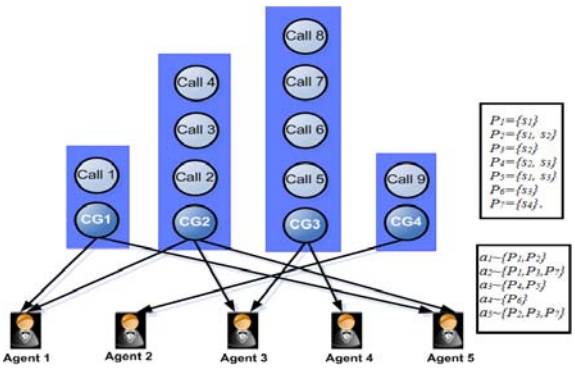

Figure 1: Inbound scheme in MSCCs with *5* agents, *9* incoming calls and *4* CGs.

The complexity of this problem is huge because we are not only dealing with an NP-hard problem like in the job assignment problem (Chauvet, 2000), but also considering high dynamism, massive incoming customer calls and large number of agents

having multiple skills. Besides, since customer calls are not planned, this makes the call assignment a truly laborious task.

3 ENCODING AND TARGET FUNCTION

In order to present a fair comparison among the analysed MHs (ILS, SA, VNS and MAs), this section provides a common problem representation and a target function to evaluate the rightness of the candidate solutions for every MH.

3.1 Encoding

The first stage when designing an MH is to define a problem representation to encode candidate solutions to the problem in a form that every computer can interpret. There are multiple forms to encode candidate solutions which range from binary strings, arrays of integers or arrays of decimal numbers to strings of letters. Specifically, our solution consists of an integer representation. We just need an array of integers whose indexes represent the available agents at a given instant and the array contents refer to the profile, P_j, assigned to each agent a_i. Then, incoming calls are "routed" to the agents, according to the profiles assigned. Of course, we can also encode the solution as an array of integers whose indexes symbolise the task types and its respective contents represent the number of agents assigned to each task type. This option is recommended whether there are too many agents and hardware capacity is very limited (with respect the total number of available agents). In contrast, we are missing the capability of working at agent's profile level. As we have not this capacity constraint, we will employ the first codification proposed.

Figure 2 shows a fictitious example (related to Figure 1) of encoding for *9* customer calls (c_1-c_9) queued in *4* different CGs (cg_1-cg_4) depending on the nature of the calls, *5* agents (a_1-a_5) and *7* profiles (P_1-P_7), where P_1={s_1}, P_2={s_1, s_2}, P_3={s_2}, P_4={s_2, s_3}, P_5={s_1, s_3}, P_6={s_3} and P_7={s_4}. Now, suppose that the agents have the following potential skill profiles: a_1~{P_1,P_2}, a_2~{P_1,P_3,P_7}, a_3~{P_4,P_5}, a_4~{P_6} and a_5~{P_2,P_3,P_7}. We have seen the potential profiles for every agent but only one profile can be assigned to each agent at a given instant *t*; therefore, a feasible solution would be Figure 2. Note that more than one agent can have assigned the same profile (e.g. a_1 and a_5).

Index (agents) →	1	2	3	4	5
Content (profiles) →	2	7	4	6	2

Figure 2: Example of the problem encoding.

3.2 Target Function

The target function is an evaluating mechanism which is defined over the encoding to measure the quality of a given candidate solution. This function often guides the search and decides which solutions must be selected for the next iteration. The target function is inherently linked to the problem. Frequently, the hardest action when defining an MH is to identify the right target function. Sporadically, it is hard (sometimes impossible) to characterise the target expression. In other cases, long evaluating times imply that an approximate function is needed. The target function that we will use is the one provided by Millán-Ruiz (2010).

4 DEFINITION OF META-HEURISTICS AND TUNING

In this section, we briefly explain some of the previously hinted MHs and their configuration for our problem. For all of them, we start from a feasible randomly generated solution.

4.1 Simulated Annealing

SA is an MH of variable environment, which generalises Monte Carlo's method (Kirkpatrick, 1983). SA proposes that the current state of a thermodynamic system is equivalent to the candidate solution in optimisation, the energy equation for a thermodynamic system is analogous to a target function, and ground state corresponds to the global minimum. This technique has the ability to hinder getting trapped in local optima since the algorithm allows for changes that decrease the values returned by the target function with a given probability (temperature). The main complexity is to determine the right value for the temperature.

Concretely, we consider Cauchy's scheme to cool off the temperature because it is faster than the Bolzmann's criterion and we only have *300* seconds to provide the system with a new solution. In Cauchy's scheme, the temperature is defined as $T_{it} = T_0/(1+it)$, where *it* is the iteration number and the initial temperature is

$T_0 = (\mu / -\log(\Phi)) * f(S^*)$ where *f(S*)* is the cost of the initial solution, Φ is the probability of accepting a "μ" worse solution than the current one (in our experiments, $\Phi = \mu = 0.3$). Finally, the maximum of neighbour solutions generated each time is *L(T)=30* and the probability of accepting a worse solution is *exp(-δ/Tit)* given that *δ=f(Neighbour_Solution)-f(Current_Solution)* and T_{it} is the temperature at iteration *it*.

4.2 Iterated Local Search

The basic idea of ILS (Stützle, 2006) is to concentrate the search on a smaller subspace defined by the solutions which are locally optimal to the current one. ILS consists of the iterative application of a more or less simple LS method. To avoid getting trapped in local optima, a perturbation is applied before executing each LS.

In our case, the complete process is repeated during *300* seconds as this is the size of the time-frame imposed by the MSCC which is being analysed. The perturbation applied after each LS affects to the *3%* of agents (the higher this perturbation is, the more random the search gets). Finally, we present the pseudo-code of an LS based on the best neighbour scheme:

```
Local_Search (candidate_solution)
{
best_solution ← candidate_solution;
neighbour ← candidate_solution;
For(i=0;i<size(candidate_solution);i++)
{
  Agent a = neighbour.getAgent(i);
  For(j=0;j<a.number_profiles();j++)
  {
    neighbour.change_profile(i,j);
    if(neighbour.fitness()>best_solutio
    n.fitness())
      best_solution = neighbour;
  }
  neighbour = best_solution;
}
candidate_solution = best_solution;}
```

4.3 Variable Neighbourhood Search

VNS is an MH whose fundamental idea is to cause systematic changes in the neighbourhood of an LS procedure (Mladenovic, 1997). VNS escapes from local optima by changing of environment *e*, increasing the size of the neighbourhood nh_e until a local minimum better than the current one is reached.

We consider three different environments $e_{max}=3$:

$$e_1 \rightarrow nh_1 = 0.3 \times n; e_2 \rightarrow nh_2 = 0.5 \times n; e_3 \rightarrow nh_3 = n$$

Similarly to the previously described techniques, these steps are repeated during *300* seconds (stopping condition).

4.4 Memetic Algorithms

MAs represent a growing research area in evolutionary computation. MAs are a variety of population-based techniques for heuristic search in optimization problems. This technique is much faster than traditional Evolutionary Algorithms (EAs) for many problem domains. Fundamentally, these combine EAs' operators with Local Search (LS) heuristics to refine candidate solutions. To design our MA, we have considered a steady-state genetic algorithm combined with the basic local search described for the ILS in Section 4.2.

Now, we briefly comment the final configuration of the evolutionary operators of our MA as the reasoning of this choice is a very significant topic and deserves to be presented in a separate study.

The configuration of the evolutionary operators is the following one:

Population: The population contains *20* different individuals.

Selection: Since the population needs to be bred each successive generation, we have chosen a binary tournament selection (Prügel-Bennett, 2000).

Crossover: The following step is to produce a new generation from selected individuals. Concretely, we consider that children will inherit the common points in their parents and randomly receive the rest of genes from them.

Mutation: This operator causes tiny changes in the genes of the chromosome to explicitly maintain diversity (actually there are much more mechanisms). In this study, we apply a perturbation of a *3%* over the chromosome.

Replacement policy: Finally, we decide which individuals are incorporated (or maybe reinserted) into the population. In this study, we consider elitism (Chakraborty, 2003) with a probability of *0.93* to replace the worst individuals of the population for next generation. And, with a probability of *0.07*, a worse individual may be captured.

Subpopulation for LS: The LS is applied over the best *25%* of individuals.

LS frequency: The LS is applied over the selected individuals each *10* generations.

Stopping condition: All steps are carried out until our *300*-second termination criterion is met.

5 EVALUATION OF RESULTS

In this section, we analyse the results achieved by all the MHs for two different problem instances (medium and high difficulty, respectively). These two instances are real data taken from our MSCC's production environment during two different days at the same hour (from *12:40* to *12:45*, *300* seconds): a one-day campaign and a normal day. The size of the time-frame to execute all the MHs is *300* seconds (*5* minutes) because we need to provide the system with a solution each *300* seconds (continuous re-planning of the ACD). We have selected this time interval because this hour (between *12:30* and *13:00*) is very representative as this is precisely the most critical hour of the day (highest load of the day: *n/m*). Note that around *800* incoming calls (*n*) simultaneously arrive during a normal day in such a time interval, whereas up to *2450* simultaneous incoming calls may arrive during this interval during a commercial campaign. The number of agents (*m*), for each time interval, oscillates between *700* and *2100*, having *16* different skills for each agent on average (*minimum=1* and *maximum=108*), grouped in profiles of *7* skills on average. The total number of CGs considered for this study is *167*. Therefore, when the workload (*n/m*) is really high, finding the right assignment among agents and incoming calls becomes fundamental. In this way, we have run every MH under two double-core processors of a *Sun Fire E4900* server (one processor for the interfaces and data pre-processing, and the other one for each MH).

Once the magnitude of our MSCC has been presented, each MH is compared alongside the others. Table 1 summarises the results obtained by each MH in *50* executions, starting from *50* different randomly generated initial solutions.

In our comparative study, we present dissimilar MHs which cover diverse strategies. Theoretically, due to the local character of the basic LS, it is complicated to reach a high-quality solution because the algorithm usually gets trapped in a neighbourhood when a local minimum is found. This occurs because the engine is always looking for better solutions which probably do not actually exist in the neighbourhood. For this reason, sometimes, it is more appropriate to allow deterioration movements in order to switch to other regions of the search space. This is precisely the shrewd policy of SA whose temperature allows for many oscillations (the probability of accepting a worse solution decreases according to the time) at the beginning of the process and only few ones at the end (fewer chances to select a worse solution as the algorithm is supposed to be refining the solution at this point). Specifically, we have chosen Cauchy's criterion because the convergence is faster than Boltzmann's and we only have *300* seconds to run the complete process. Besides, this scheme avoids decreasing the distance between two solutions when the process converges (jumps in the neighbourhood). Therefore, the temperature must be high enough at the beginning to better explore the search space (its neighbourhood) and low enough at the end to intensify the search as well (exploitation of promising areas). The value for speed is, therefore, the stopping condition which must agree with the number of neighbours generated.

Table 1 gathers the results obtained by each MH in *50* different executions for two different problem instances with the purpose of providing a fair comparison. The first three columns are the best, worst and mean fitness values, respectively. Then, we have the standard deviation and the effectiveness (best fitted solution represents the *100%*).

We perceive from Table 1 that SA worse behaves than the other MHs except for the easiest instance of the problem. This may occur because we are not plenty of time in our environment and the power of SA relies on a progressive cooling. If we cool off the temperature too fast, we are missing the effectiveness of accepting worse solutions in some cases. Instead, if we cool off the temperature too slowly, we may be accepting worse solutions systematically without converging. We have applied a trade-off between exploration and exploitation but the time seems to be limited to apply SA to our environment (perhaps, things might change when having more time).

Another option to increase the diversity in the solutions is to enlarge the environment, as VNS does. This philosophy consists of making a systematic change upon the environment when the LS is used, increasing the environment when the process becomes stagnated. In the VNS, the search is not restricted to only one environment as in the basic LS; instead, the neighbourhood changes as the algorithm progresses. Albeit we only consider three distinct neighbourhoods, the improvement of the VNS compared to basic LS is noteworthy. Consequently, the remarkable factor becomes the change in the number of neighbourhoods and their sizes as well as to consider how the algorithm reacts in response.

Table 1 also shows how VNS only slightly outperforms SA for the hardest instance of the problem.

Another strategy is to start from different initial solutions as ILS accomplishes. ILS generates a

random initial solution and afterwards applies a basic LS. Subsequently, this solution is systematically mutated and thus refined. ILS obtains solutions which vaguely improve those given by SA and VNS for the hardest problem instance, although it performs worse for the simplest problem instance as Table 1 corroborates.

Another way to find a solution involves using methods based on populations, such as MAs. If the diversity of the solution is low, then the MA converges to the closest neighbour. Nevertheless, when the selective pressure is high, individuals may be alike or even identical. To speed-up the convergence, MAs apply an LS upon a set of chromosomes (candidate solutions) that are refined every certain number of generations. Incorporating a hybridisation mechanism to the GA is valuable as the algorithm is improved in all respects. This fact is pointed up in Table 1 as the MA not only outperforms all the strategies for both instances but also remains more unwavering (less differences among best, worst and mean fitness values).

Finally, it is important to remark that differences among techniques are not huge after reaching a fitness of *0.8* since the complexity increases exponentially in our environment. Therefore, minor improvements on the fitness value after that point are hard to obtain but very valuable to accomplish a fair workforce distribution.

6 CONCLUSIONS AND FUTURE WORK

We have seen the difficulty of assigning incoming calls to the right agents within a real-world MSCC. We have then presented several MHs to carry out this task and how MAs can outperform them in such an environment. Afterwards, we have done an exhaustive comparative study of MHs to empirically evaluate diverse strategies to reach accurate, feasible solutions, capturing real data from an MSCC within a large multinational telephone operator during a peak of load (*12:40-12:45*). This study has illustrated how these MHs perform for two different problem instances. We can conclude that MAs not only outperform all the strategies but also remain more unwavering as systematically provide better results than the rest of techniques. As future work, we propose to do a similar study considering parallel MHs and different time-frame sizes.

REFERENCES

Chakraborty, B. and Chaudhuri, P., 2003. *On The Use of Genetic Algorithm with Elitism in Robust and Nonparametric Multivariate Analysis*, Austrian Journal of statistics.

Chauvet, F.; Proth, J. M.; Soumare, A., 2000. *The simple and multiple job assignment problems*, International Journal of Production Research, Volume 38, Issue 14, pages 3165-3179, 2000.

Garnett O. and Mandelbaum, A., 2000. *An Introduction to Skills-Based Routing and its Operational Complexities*, Teaching Note.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983. *Optimization by Simulated Annealing*, Science, Volume 220, Number 4598, pp. 671680.

Koole, G., 2006. *Call Center Mathematics: A scientific method for understanding and improving contact centers*, http://www.cs.vu.nl/~koole/ccmath/book.pdf, 2006.

Millán-Ruiz, D. and Hidalgo, I., 2010. *A Memetic Algorithm for Workforce Distribution in Dynamic Multi-Skil Call Centres*, EVOCOP 2010, pp. 178-189.

Mladenovic, N. and Hansen, P., 1997. *Variable Neighborhood Search*. Computers & Operations Research 24, pp. 1097–1100.

Prügel-Bennett, A., 2000. *Finite Population Effects for Ranking and Tournament Selection*. Complex Systems, 12 (2), pp. 183-205.

Stützle, T., 2006. *Iterated local search for the quadratic assignment problem*. European Journal of Operational Research, Volume 174, Issue 3, pp. 1519-1539.

Table 1: Results obtained by the MHs in *50* executions starting from random initial solutions for two problem instances: medium and hard (larger number of incoming calls and high variability). Values refer to the fitness obtained by all MHs.

Algorithm	Best solution		Worst Solution		Average		Standard deviation		Effectiveness	
	Medium	Hard	Medium	Hard	Medium	Hard	Medium	Hard	Medium	Hard
MA	0.796	0.758	0.785	0.751	0.796	0.754	0.001	0.001	100	100
ILS	0.768	0.728	0.755	0.722	0.763	0.725	0.002	0.003	95.85	96.15
VNS	0.790	0.727	0.766	0.723	0.775	0.724	0.005	0.001	97.36	96.02
SA	0.782	0.721	0.773	0.709	0.779	0.716	0.001	0.003	97.86	94.96