A landscape analysis for a Hybrid Approximate Algorithm on a Timetabling Problem

Marco Chiarandini and Thomas Stützle
TU Darmstadt, Computer Science, Intellectis Group
Hochschulstr. 10, 64289 Darmstadt, Germany
{machud|stuetzle}@intellektik.informatik.tu-darmstadt.de

The university course timetabling problem is an over-constrained optimisation problem that consists in scheduling a set of events in a given number of rooms and timeslots. In a previous work a general case of the problem was considered and an effective hybrid algorithm based on local search optimisation was developed to solve it. Landscape analysis is the new trend to explain local improvements algorithms providing validations or refutations to possible insights on the algorithm behaviour. Usually used on problems without too many technicalities like Satisfiability Problem, Quadratic Assignment Problem and Scheduling, it is not clear if landscape analysis can help also on over-constrained problems, like the case of the timetabling problem. In this work we report about our attempt in this area of research.

1 Introduction

The university course timetabling problem (UCTP) consists in assigning a set of lectures to rooms and timeslots subject to a number of hard and soft constraints. Violations of hard constraints make the assignment infeasible while failing to satisfy soft constraints induces some penalty cost that has to be minimised.

In a previous work we developed an effective algorithm based on local search optimisation for solving (approximately) a general formulation of UCTP. The algorithm ranked the best among several other candidates and it is therefore a state-of-art algorithm for UCTP. A comprehensive description of the algorithm is in [1]. This research is the outcome of the work done in the context of the Metaheuristics Network, a Research and Training Network funded by the Improving Human Potential programme of the European Commission. The main goal of the Metaheuristics Network is to improve the understanding of how metaheuristics work with theoretical and experimental research.[1]

Local search to solve combinatorial optimisation problems is an empirical method accepted in the scientific community because it succeed where mathematical models appear

[1]More information on the Metaheuristics Network research programme is available at `http://www.metaheuristics.org`

difficult. Currently undergoing research in the attempt to explain and understand its behaviour are based on empirical theories. Yet a clear theory has not arisen. Conjectures that deals with the topology of the search space appear rather weak at the present time and vary often from problem to problem. Landscape analysis is the field where these conjectures are tested in order to be verified or falsified.

Like for every theory, the final aim of landscape analysis is understanding and predicting the specific behaviour of local improvement algorithms in a given problem. The landscape analysis provides experimental arguments for the examination and, in case, validation of some guidelines that can be recognised in local search.

In this work, we look for models that account for the hardness of the instances of a specific problem based on some features of the search space defined by the algorithm in use. We test them experimentally and, in case, accept them or adjust them towards other that are better verified from empirical data. Successful examples of this way of proceeding arise in the field of the Satisfiability Problem [2], the Quadratic Assignment Problem [3] and the Job-Shop Scheduling [4].

Nevertheless, given the complexity of the problems to be solved by local search algorithms, landscape analysis is not an easy task, and often simplifying assumptions in the analysis are needed.

In the case of over constrained problems and more specifically of timetabling problems several difficulties arise when attempting to describe the features of the search space. In this work we point out these difficulties and we make some assumptions to overcome them. The final analysis can only be approximated and answer cannot be definitive, yet we test some well known models and see that very few positive indications are found.

In the following, first we introduce the problem and the class of instances to be solved in Section 2, and we briefly present the algorithm under study in Section 3. Then, in Section 4 we discuss the main models accounting for some behaviour of the algorithm and test them with the experimental results produced by the landscape analysis. We conclude resuming the main results in Section 5.

2 The UCTP and the class of instances

The UCTP problem we studied is a realistic reduction of a real life case. It consists of a set of events or classes E to be scheduled in 45 timeslots (5 days of 9 slots each), a set of rooms in which events can take place, a set of students who attend the events, and a set of features satisfied by rooms and required by events. Each student attends a number of events and each room has a size. A feasible timetable is one in which all events are assigned a timeslot and a room so that the following *hard constraints* are satisfied:

- no student attends more than one event at the same time;
- the room is big enough for hosting all attending students and satisfies all the features required by the event;
- only one event is assigned to each room at any timeslot.

In addition, a candidate timetable receives a penalty cost for violating any of the following *soft constraints*:

- a student has a class in the last slot of a day;
- a student has more than two classes in a row;

Procedure 1 Hybrid Algorithm UCTP Solver.

```
Input: A problem instance $I$
  for i=1 to $max\_heur$ do
    $s_i \leftarrow$ BuildAssignment
    $s_i \leftarrow$ HardConstraintsSolver($s_i$)
    $s_i \leftarrow$ LookAheadLocalSearch($s_i$)
  end for
  $s_{best} \leftarrow$ SelectBestAssignment($S$)
  $s_{best} \leftarrow$ SoftConstraintsOptimiser($s_{best}$)
Output: An optimised assignment $s_{best}$ for $I$
```

- a student has a single class on a day.

The objective is to find a feasible assignment that minimises the number of soft constraint violations. An infeasible assignment is considered worthless. Here we make indistinct use of the terms *timetable*, *assignment* and *solution* to indicate a complete placement of events to timeslots and rooms independently from the fact that it is feasible or not. Furthermore, with the expressions *number of soft constraint violations* and *cost of a solution* we mean the total penalty cost due to the soft constraints not satisfied by the assignment. For a detailed definition on how these penalties costs are computed we refer to [1].

The instances we use for our analysis are the same 20 instances used for ranking different algorithms tested on the UCTP. They were created generated by a random generator which also provide an optimal assignment for each instance generated. The instances present between 350 and 440 events, between 200 and 300 students, and around 10 rooms. The number of timeslots is fixed to 45.

Since it is known that zero violations is always possible, assignments exist that use only 40 timeslots (due to the soft constraint that forbids the use of the last slot of a day). With 400 events and 10 rooms this corresponds to fill up all timeslots with events. Therefore, solving the instances proposed, should apparently not be an easy task.

3 The algorithm adopted

The algorithm we want to study is an hybrid algorithm that works as schematically reproduced in Procedure 1. First a group of initial solutions is constructed by BuildAssignment, made feasible by HardConstraintsSolver and assessed by LookAheadLocalSearch. Then, the best assignment is selected by SelectBestAssignment and exploited by means of SoftConstraintsOptimiser.

HardConstraintsSolver and SoftConstraintsOptimiser are two procedures based on local search that deal with hard and soft constraints respectively. When looking for feasibility, soft constraints are not taken into account, while when minimising the number of soft constraint violations, hard constraints cannot be broken. SoftConstraintsOptimiser is entirely based on local optimisation where the objective function to minimise is the number of soft constraint violations. It uses before a Variable Neighbourhood strategy to descent rapidly in a local optimum, then for the rest of the time the optimisation is carried over by Simulated Annealing.

Reaching feasibility does not appear a task too difficult and in all cases, HardConstraintsSolver reaches feasibility relatively fast. Instead, finding the optimal solution with respect to the soft constraints is a much harder task. Only on 4 instances our algorithm succeeded in finding an optimal solution in reasonable time. For this reason, we focus on the second part of the algorithm where, after the best assignment has been chosen, soft constraint violations are minimised and feasibility is maintained.

The solution representation in this phase is defined by a rectangular matrix, such that rows represent rooms and columns represent timeslots. A number, corresponding to one of the events or to none of them, is assigned to each cell of the matrix. With this representation, we assure that there will be no more than one event assigned to each room in any timeslot, which means that one of the hard constraints will always be satisfied.

For the local search we defined four different relations of Neighbourhood given respectively by the move of one event, the swap between two events, the exchange between events belonging to two different timeslots and the Kempe chains. Again we refer to [1] for details. The assignment of rooms that represents the rows of the matrix can be decided by some local search strategy but it also possible, once the destination timeslot has been chosen, to assign them by mean of an exact matching algorithm. In SoftConstraintsOptimiser both methods are used. In particular, in the Simulated Annealing phase rooms are assigned by means of the matching algorithm, while in the Variable Neighbourhood phase an alternation of the two methods is applied since the matching algorithm enlarge slightly the neighbourhood but it does that in trade of an increased computation time.

4 Landscape analysis

Search space topology can reveal the reasons why some instances are harder than others. No clear model has been so far developed for linking features of the search landscape with the difficulty of solving given instances of a problem. Two widely reported measure of landscape are the *correlation length* derived by the correlation function of a random walk through the fitness landscape and the *fitness distance correlation* that determines how closely related are fitness of solutions and distance to the nearest optimum in the search space [5]. The terminology *fitness* to allude to the quality of a solution is borrowed from the field of evolutionary biology were landscape analysis was first introduced [6].

However, correlation length, for a large number of problems, is strictly a function of the problem size (*e.g.* the number of cities in the Travelling Salesman Problem) [7]. Consequently, it is unclear how correlation length can explain variance in problem difficulty when the instances are all of the same size. Fitness distance correlation, on the other side, is never enough to account for differences in the difficulty of individual instances, furthermore discordant comments on its interpretation are reported in the literature.

A set of other search space features are, therefore, usually analysed: the number of optimal solutions, the backbone size (part of solutions that are repeatedly present during the search), the average distance between random local optima, and the mean distance between random locally optimal solutions and the nearest optimal solution. These features are put in relation with the cost of finding an optimal solution for each given instance in order to validate or falsify conjectured relations [4]. Analysis of search space topology considering these features have been done on Satisfiability Problems [2] and Job-Shop Scheduling Problems [4] while to our knowledge, no attempt was made for the timetabling problems.

4.1 Landscape Analysis for the UCTP

In our specific UCTP some difficulties arise when investigating for the search space features introduced in the previous paragraph. Firstly, in general only one optimal solution for each instance is available but no knowledge is given about how many optimal solutions

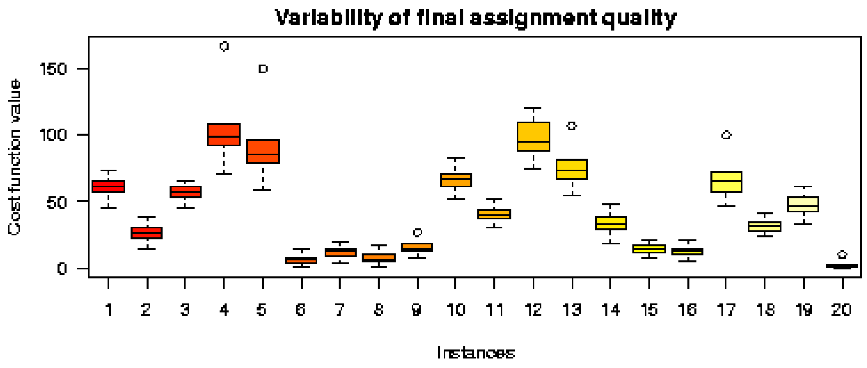

Figure 1: The box-plots summarise the costs of solutions found by 50 trials on each instance. The central box shows the data between the quartiles, the median is represented by a line in the central box. "Whiskers" extend to data points which are no more than 1.5 times the interquartile range away from the box. Circles represent very extreme data.

there could be and an exact algorithm for solving this timetabling problem is unlikely to be developed, given that it is an $\mathcal{NP}$-hard problem and that the size of the instances is large. Secondly, given the hybrid nature of our algorithm, the definition of the two basic measures, the *cost* of finding an optimal solution and the *distance* between different assignments in the search space, is a complex task.

For *cost of an instance* we assumed the median value of the costs distribution of 50 runs per instance. The box-plots of these distributions are reported in Figure 1. Usual measures that account better for the the cost of an instance are the number of iterations or the time needed to reach the optimal solution or a given quality solution. These methods are not feasible in our case since defining what an elementary step is is not possible in the general case of different neighbourhoods. Moreover, we do not reach always the optimal solution and defining a quality poses the problem related with cost normalisation. But the normalisation of the cost requires to know the maximum value of the evaluation function which is here not an easy task.

For *distance* between two assignments, defined as the minimum number of elementary transformations necessary to transform an assignment into another, we used the 1-opt distance, that is, we confined ourself to compute only the distance defined by moves of single events into other timeslots. This distance is a surrogate since the transformation depends on the neighbourhood currently adopted and in an algorithm that varies neighbourhoods during the search process, an unique definition of distance is not possible. Beside this, we assumed another main simplification. We ignored room assignment. In some procedures our hybrid algorithm uses a deterministic procedure. In others we simply suppose it is not the main cause for two assignments to be different, and that adjusting events and timeslot can be enough to pass from one assignment to another.

Important to be noticed is instead the fact that for how we evaluate an assignment, there is no difference between assignments with the days permuted. In this case the distance is more precisely referred as *partition distance*. Keeping this in mind, we computed the distance between two assignments A and B in the following way. For each day of assignment A we calculate the minimum number of transformations to bring it to coincide with each day of assignment B and we store these values in a 5×5 matrix called matrix of costs. Next, we look for the permutation of days in the assignment B that would yield the minimum global number of transformations to go from assignment A to the permuted assignment B. This, as noticed in the case of the distance between assignments of graph colouring, coincides with solving a linear assignment problem on the matrix of costs [8]. For this task we used the algorithm developed by Jonker et al. [9] that does not solve the linear assignment problem in polynomial time but it is, nevertheless, very fast for the small size of matching instances we have to consider (one of the two sets of the bipartite graph to solve for the matching never exceeds 11 nodes which is the maximum number of rooms available).[2] It remains to be said how we compute the minimum number of transformation to go from whichever day of assignment A to whichever day of assignment B. We do a matched pair comparison between timeslots, *i.e.* between timeslots with the same index in the two days, and we count the number of events which are in the timeslot of the assignment A but not in the correspondent timeslot of the assignment B.

To sum up, the way we compute the distance is affected by the following approximations: (i) it considers only moves of events into another timeslot while as we described in Section 3 the neighbourhoods we used extend the variety of available moves; (ii) it does not consider rooms; (iii) it is done in a search space that includes also infeasible assignments while our algorithm does not allow to exit from feasibility, hence, the path with minimum distance can actually be banned. Given these considerations, the outcome of the landscape analysis must be taken with caution.

In the following we investigate the search space features and quantify the accuracy of the proposed models using linear regression techniques.

4.2 Quality of local optima

The first feature that we considered is the quality of local optima. Local search tries to find improved solutions by considering neighbours of a current solution. By definition a local optimum is the solution for which no neighbour assignment can be found that is better. Intuitively, the worse the quality of local optima the harder the problem should be. In our case, we define local optimum the assignment reached after applying Variable Neighbourhood Search in all the four neighbourhoods, as defined in Section 3. Actually a local optimum can be part of a plateau at the bottom of a valley or it can be part of a bench, that is a plateau with some states on the border that have a better objective function [10]. Our implementation of Variable Neighbourhood Search scans all four neighbourhoods in a first improvement strategy but it accepts also side walk moves (moves which leave the quality of a solution unchanged) with a probabilistic criterion, in order, if possible, to move through the plateau in search of an exit, that is a border with better objective function. When the four neighbourhoods have been entirely scanned and during the scan no improvement has been found, we stop the search and consider the last assignment as a local optimum. Side walk moves are performed during the scan in accordance to a given probabilistic criterion but they are not considered as improvements.

For each instance we generated 200 local optima and we considered the medians of the distribution of their quality. In Figure 2 we show scatter plots of the median quality of local optima versus the instance cost. The r^2 value for the corresponding regression model is 0.587 and the F-test provides statistical significance to reject the hypothesis that the two measures are independent. According to this result, we accept the model suggested by intuition. Furthermore it is worth noticing that it is in principle possible early in the search to predict which will be approximately the final solution.

[2]Available via `http://www.magiclogic.com/assignment.html`, June 2003

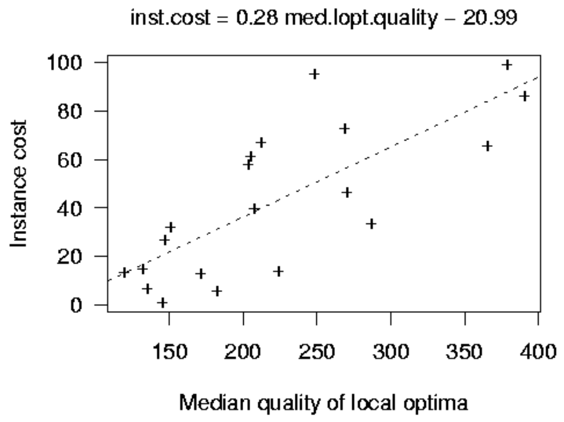

Figure 2: Scatter plot of the median costs of local optima versus the cost of an instance (median costs of final solutions); a least-squares fit line is superimposed. The title of the plot reports the regression coefficients of the fitted line. The corresponding r^2 value is 0.587.

We investigated also the correlation between the quality of the initial feasible assignment chosen and the final solution quality. A correlation between these two features would indicate a certain importance of starting from a good assignment and could justify the selection of the best candidate among an initial set of candidates. But in this case no meaningful model correlating the two measures was found. This fact suggest a possible improvement to the algorithm. Indeed, it tell us that what matters in terms of final solution is the selection of the candidate assignment that produce the best local optimum. The current algorithm, instead, makes its choice before a local optimum has been reached. Unfortunately, reaching a local optimum for all candidates implies longer computation time and it has to be understood if the trade off with the number of candidates can be positively solved.

4.3 Minimum distance between random local optima

Search algorithms are strongly biased by local optima. Consequently features of the sub-space of local optima could be somehow correlated with the search cost [4]. We considered the minimum distance between distinct pairs of random local optima normalised by the number of events. We kept this as an indicator of the size of the sub-space of local optima and we considered whether differences of this value account for differences in the search cost. For this purpose we generated 200 local optima per instance, for each of them we took the minimum normalised distance from all other local optima and we, then, derived the average value. Figure 3 reports our attempt to model the instance cost with the average minimum distance between local optima. The r^2 is very small, only 0.115, and statistical tests suggest that independence of the two measures can not be rejected.

It is worth noting, however, that the distance between local optima is very large.

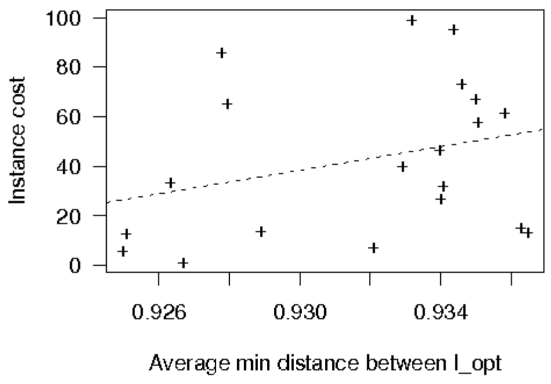

Figure 3: Scatter plot of the average minimum distance between local optima (l_opt) versus the cost of an instance; a least-squares fit line is superimposed. The corresponding r^2 is 0.091. Instances considered are the 20 of the competition. 200 local optima per instance are considered.

4.4 Minimum distance between local optima and global optimum

Another important feature of the landscape is the distance between local optima and nearest global optima. In the Job-Shop Scheduling Problem this indicator accounted quite well for differences in the search cost [4]. Since we do not know all global optima, we limit ourselves to consider the minimum distance of local optima from the known global optimum, that is, we search in an heuristic way around the plateau of the global optimum for finding the assignment with smallest distance. To move in the plateau we used again a search heuristic, precisely a modification of our local search in the events swap and exchange neighbourhoods that minimises distance while keeping fixed the cost value. We enhanced this procedure in order to avoid repetition of visited assignments by means of Tabu Search. No significant dependence arose between the average of the minimum distances of 200 local optima per instance and the cost of the instance, therefore, the model fails to account for differences in the cost of an instance. In Figure 4 we plot the results. There is almost no correlation.

For each instance we also considered the relation between the minimum distance of local optima from global optimum and the quality of local optima. Intuitively assignment closer to optima should have better quality, but this was not confirmed by our experiments, where no significance to reject the hypothesis that the two measures are independent arose. This result is however weak since we know only one global optimum, and the way we compute the distance may have an influence.

4.5 Number of improvements needed to reach local optima

The effort spent to reach local optima can also give an indication on the topology of the search space. We considered the SoftConstraintsOptimiser without Simulated Annealing and we counted the number of local search used but no significant incidence of the number of local searches performed was found on the quality of local optima and no significant correlation arose either with the cost of the instance.

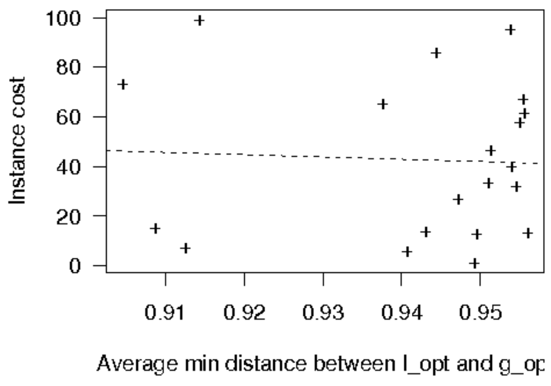

Figure 4: Scatter plot of the minimum distance of local optima (l_opt) from global optima (g_opt) versus the cost of an instance; a least-squares fit line is superimposed. The corresponding correlation value is 0.003. Instances considered are the 20 of the competition and 200 local optima are used.

We then considered the number of side walk moves accepted and compared it against the number of improving moves. The number of side walk moves was on average 9 times bigger than the number of improving moves. A correlation value of 0.739 between them indicates that the amount of improving steps increases when side walk moves are possible. Side walk moves are however correlated with the size of plateaus in the instances (see next paragraph). Instances with larger plateaus have also higher number of side walk moves on average.

4.6 Plateaus size in local optima and global optima

To explore the size of plateaus we developed a Tabu Search procedure that tries as much as possible to modify an assignment without changing its evaluation function. In order to avoid visiting twice the same assignment, it keeps an archive of the complete representation of visited assignments. The management of such a list of visited assignment is costly, and consequently our experiments were limited. In the right part of Figure 6 we report the bar plots of the size of plateaus found in a time that was about 10 times longer than the time we usually used to solve the instances. In some instances this time was not even enough to complete the exploration of the plateaus around one single local optimum. In other instances, where more than one plateaus was explored, we consider the largest found. With some exceptions the size of plateaus is very large. The largest is 200 thousand assignments and time expired before the search ended. In some instances, however, plateaus are small. Surprisingly when this occurs for several local optima it occurs also with the global optimum. Consequently the size of plateaus around local optima seems to be a feature of the instance. We investigated models that try to deduce the size of plateaus by some characteristics of the instance but no significant correlation was found. Finally, it is also worth to mention the absence of correlation between size of plateaus and final solution qualities.

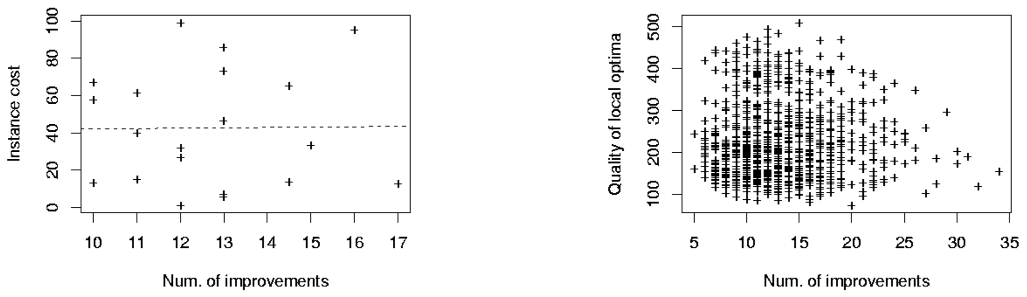

Figure 5: On the left, scatter plot of the median number of improvements per instance versus the cost of an instance; a least-squares fit line is superimposed. The corresponding correlation value is 0.009. On the right, scatter plot of the number of improvements versus the cost of assignment attained. The correlation is 0.275. 50 local optima per instance are considered in both the graphs.

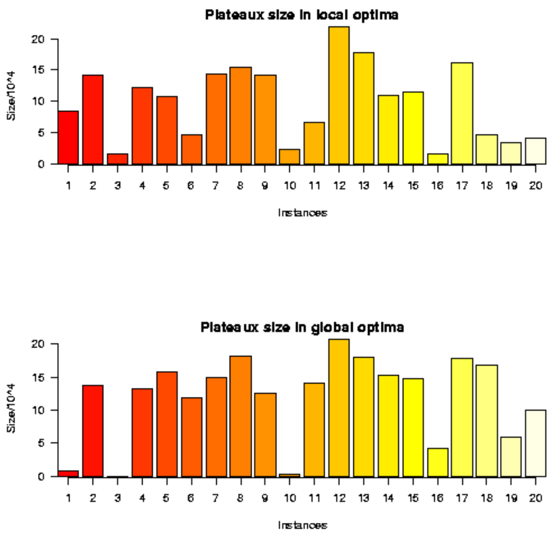

Figure 6: Size of plateaus in local optima (upper part) and in global optima (lower part). Plateaus are explored with a local search that avoids to repeat states already visited. The time limit was set about 10 times bigger than the time allowed to solve the instance. In some cases time was not enough to finish to explore one single plateaus. In other cases more than one plateaus was visited and we report the size of the larger found.

4.7 Connectivity of the search space

Another interesting issue for highly constrained problems is that feasible areas of the search space may be disconnected. This is a strong drawback for search heuristics which could remain trapped in a non promising region but this seems not to be the case with the instances under examination. Indeed, given that an optimal solution does exist, there are always free slots that can be used to move from one solution to another without breaking hard constraints. The only hard constraint that could be violated would be assigning two correlated events to the same timeslot, but this appears unlikely to be the only chance to reach the destination assignment. Therefore, the landscape appears to be connected even if it can not always be possible to walk through the path of minimum distance. This intuition was confirmed by an experiment in which we tried to move from a global optimum to another using a local search intended to minimise distance without breaking feasibility instead of the evaluation function. The experiment succeeded and it tell us that we can not reject the conjecture that the search space is connected. However we must recognise that the test was not stringent since the experiment was limited only to instance 20, for which we had available more than one global optimum.

5 Conclusions

The empirical analysis presented in this report confirmed only few models to locate good quality assignments in the search space. We found confirmation that the quality of the local optima has an influence on the final solution quality. We had to reject instead some intuitions, like that good local optima are grouped in a specific region of the search space or that the closest the local optima are to the global optimum the best they are. We did not find support either for another conjecture, actually useful when developing the algorithm, that side walk moves are profitable and that the search space presents always large plateaus. It is true that side walk moves help in getting better results because these was revealed by the experiments in the development phase and it is also true that plateaus are often very large. But it is not possible to understand directly the influence of plateaus or side walk moves on the quality of the final assignment. In some cases the size of plateaus is much smaller but this seems not to happen in correlation with the hardness of an instance.

We failed instead to reject the conjecture that the search space is connected. If we accept this conjecture we can conclude that local search algorithms that do not break feasibility once it has been reached can work well, and with a good mechanism to get out of local optima and sufficient time they can even reach the global optima. Yet, if – like in our case – reproducing feasibility is very costly in terms of soft constraints, those algorithms are clearly preferable. However the same conclusion was suggested already by the experimental methodology adopted to select and to tune the algorithm, where algorithmic configurations that allowed to exit from feasible regions were rejected because they performed worse than configurations that did not exit from feasibility.

Acknowledgements The authors want to thank Mauro Birattari, Krzysztof Socha, and Olivia Rossi-Doria for contributing in developing the algorithm under study and Tommaso Schiavinotto for discussions on the topic of landscape analysis. This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid approach for the university course timetabling problem. Submitted to journal, April 2003.

[2] D. A. Clark, J. Frank, I. P. Gent, E. MacIntyre, N. Tomov, and T. Walsh. Local search and the number of solutions. In *Principles and Practice of Constraint Programming*, pages 119–133, 1996.

[3] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, November 2000.

[4] J. P. Watson, J. C. Beck, A. E. Howe, and L. D. Whitley. Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence*, 143(2):189–217, February 2003. Preprint.

[5] P. Merz and B. Freisleben. Fitness landscapes and memetic algorithm design. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 245–260. McGraw-Hill, London, 1999.

[6] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc. of the Sixth Congress on Genetics*, page 365, 1932.

[7] P. F. Stadler. Landscapes and their correlation functions. *J. Math. Chem.*, 20:1–45, 1996.

[8] D. Gusfield. Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82(3):159–164, May 2002.

[9] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[10] J. Frank, P. Cheeseman, and J. Stutz. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.