Experimental Evaluation of Course Timetabling Algorithms

Marco Chiarandini and Thomas Stützle
Fachgebiet Intellektik
Fachbereich Informatik
Technische Universität Darmstadt
Darmstadt, Germany
{machud,stuetzle}@intellektik.informatik.tu-darmstadt.de

April 2002

1 Introduction

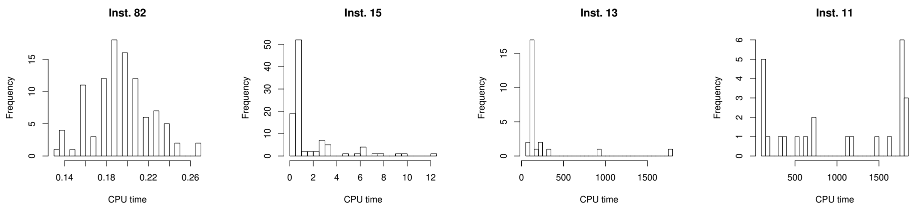
The goal of this report is to give an analysis of the run time distributions for the algorithms considered in the experimental evaluation of the Course Timetabling problem [8]. Furthermore we investigate the hardness of the instances produced by the random instance generator proposed by Rossi-Doria and Ben Paechter [8]. This analysis confirms the impression, gained during the configuration of the metaheuristics Simulated Annealing and Iterated Local Search at the Intellectics Group and suggested by the results on the SAT problem [11, 3, 10], that some instances can be substantially harder than others. By further studying the correlation of the hardness of the instances for several algorithms, we can deduce that the hardness seems to be an intrinsic aspect of an instance and does not only depend on the specific algorithm used to solve it. In addition to this, we study how the hardness of instances is influenced by some of the parameters of the instance generator.

All analyses are done on a simplification of the problem in which only hard constraints are considered and the time computed is the time needed for finding a first feasible solution. The algorithms analyzed are Iterated Local Search (ILS), Ant Colony Optimization (ACO), Random Restart Local Search (RR.LS) and Simulated Annealing. For the latter one we considered two variants: the winner of the automated tuning [1] (SA) and another variant which uses the Simulated Annealing criterion embedded in the local search described in [9] (SA.LS). More specifically, the first approach uses a completely random move selection strategy in which at each step the proposed move is randomly selected from the union of all the possible moves, while the second approach searches, in the same order as in the local search, only a promising restricted part of the neighborhood. We refer to [9] for a detailed description of the neighborhood structure used by all the algorithms.

2 Peak Performance Comparison

2.1 Hardness distributions

Assuming that all algorithms have optimal parameter settings, we want to represent the behavior of each algorithm on a set of instances. As an indicator of the behavior of an algorithm on a single instance we choose the median cost observed when trying to solve an instance in multiple trials. The median was chosen because of the large variation of the solution cost and because of the existence of pronounced tails in the distributions. Some sample distributions of the costs found in the case of Simulated Annealing on small and medium size instances are plotted in Figure 1.



Instance	Min.	1st Quartile	Median	Mean	3rd Quartile	Max.
small "82"	0.1300	0.1800	0.1950	0.1958	0.2100	0.2700
small "15"	0.3800	0.5275	0.6300	1.6890	1.9850	12.0800
medium "13"	97.29	107.00	119.80	234.00	140.90	1800.00
medium "11"	104.7	305.6	1106.0	1024.0	1800.0	1800.0

Figure 1: Histogram of 100 runs on small size instances "82" and "15" and of 25 runs on medium size instances "13" and "11" (from left to right). The table gives a summary of some basic statistics. The algorithm used is Simulated Annealing. The maximum run time allowed was 1000 seconds for small size instances and 1800 seconds for the medium size.

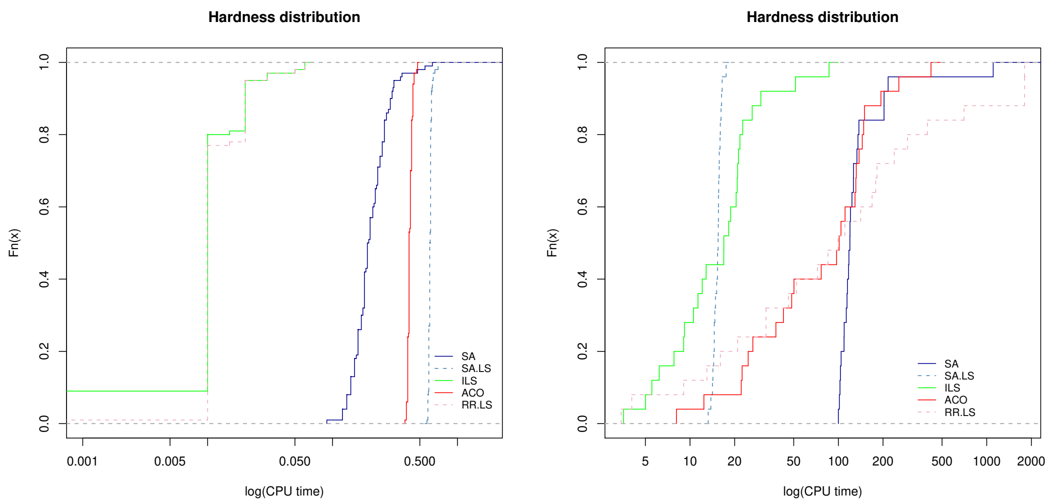


Figure 2: Hardness distributions for SA, SA.LS, ILS, ACO and RR.LS on the small size instance test-set (left) and the medium size instance test-set (right). The x axis gives the CPU time in logarithmic scale and the y axis gives the empirical cumulative distribution function $Fn(x)$ representing the probability of finding a solution at a fixed amount of time.

The hardness distributions we obtain are shown in Figure 2. For each instance the median search cost (CPU time) is measured and given is the cumulative hardness distribution for each algorithm. They are produced by 100 trials on 100 instances from a test-set of small size instances (Fig. 2, left) and by 25 trials on 25 instances from a test-set of medium size instances (Fig. 2, right). The difference between instances of the same set is solely the random seed used for generating them while the other parameters of the instance generator are the same proposed for the experimental evaluation. The maximum CPU time allowed for each run was 1000 seconds for the case of small instances and 1800 seconds for the medium one on a machine with a Pentium III 700 MHz processor with 256 KB of cache.

The following observations can be made:

- On the medium size instances RR.LS presents large *variability* indicated by the distance between the minimum and the maximum time found for solving the instances. Moreover a bunch of "hard" instances are never solved in 1800 seconds. Hence, the other algorithms appear to be preferable over RR.LS.

- The variability in computation time required for finding a feasible solution is negligible for the small size but its relevant for the medium size. There, for instance, ACO spans from less than 10 seconds on the easiest instances to almost 500 seconds on the hardest ones, meeting a difference by a factor of 50. For RR.LS the variability on the medium size instances is even stronger.

- The CPU time ACO needs for solving the small instances seems almost not to vary. This is due to the fact that the algorithm finds regularly a feasible solution right after the construction of a first population and the application of local search exclusively to its best individual. But this is not anymore true for the medium size instances where ACO shows much larger variability.

- Except for ACO on small size instances, there is a considerable variability in search cost between the instances of each test-set. This is already present in the small size set but it is confirmed and emphasized in the medium size test set. Only SA.LS does not show this behavior. This means that the use of a Simulated Annealing criterion embedded in the local search allows to find feasible solution w.r.t. the hard constraints in almost always the same time. It is an open question why this is the case even if we discovered that the best configuration for this algorithm is keeping a constant low temperature during all the search.

- The plots do not help in defining clearly which algorithm performs better. Cross-overs in the empirical hardness distributions indicate that performance depends on the instances solved. ILS, however, seems to always dominate SA (w.r.t. solving hard constraints) while SA.LS dominates SA on the medium size test set but not on the small size one.

- A small group of instances is considerably harder than others. All algorithm distributions confirm this. The most extreme case is that of RR.LS, which for some medium size instances was not able to solve them within the given time limit of 1800 seconds. The presence of instances harder than others in the single test-set is indicated by the presence of tails in the distributions. Therefore random seeds in the instances generator are already enough for producing different hardness in the instances.

2.2 Run time distributions

We now consider the cumulative empirical distributions of the time needed by algorithms for solving single instances. These *run time distributions* (RTD), which for a given, fixed instance map the run time t to the probability of finding a feasible solution within time t [7, 6], are given for the easiest, median and hardest instance of the hardness distributions of Figure 2. Figure 3 gives the corresponding plots for the small size instances and Figure 4 those for the medium size ones. The results are produced by executing 100 runs on the small instances and 25 on the medium size instances. The graph is logarithmic in the time axis.

We can observe that:

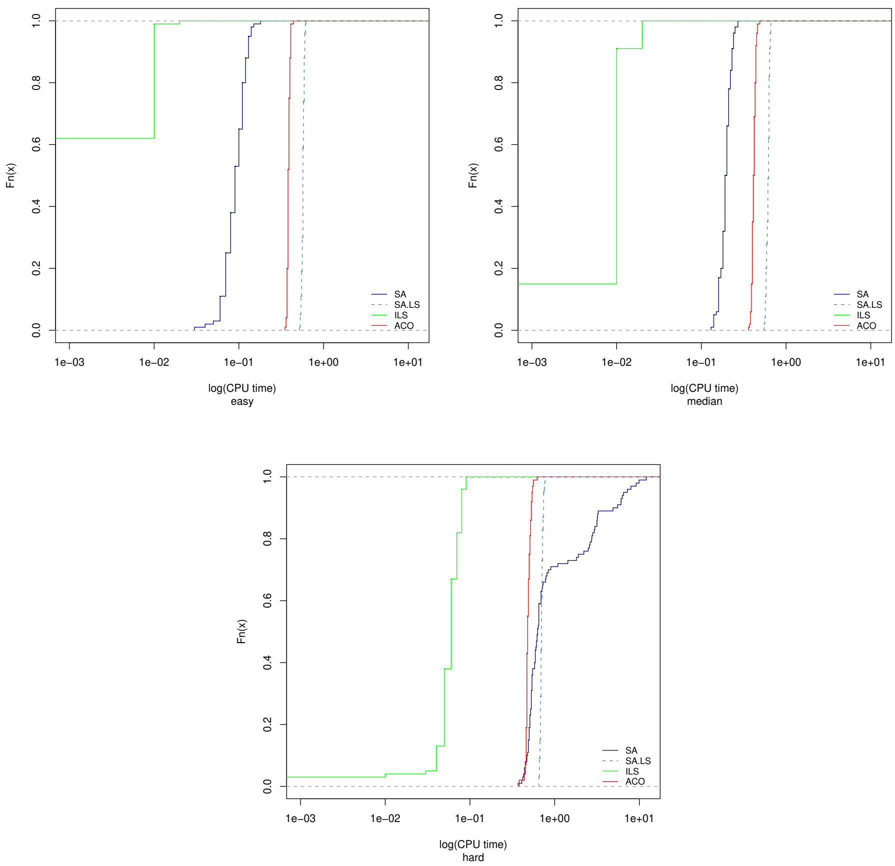


Figure 3: RTDs of algorithms on the easiest (easy), the median (median) and the hardest (hard) instance of the *small* size test set, based on 100 tries per instance. The x axis gives the logarithm of the CPU time, the y axis the empirical cumulative distribution function $Fn(x)$. The behavior of RR.LS is similar to ILS and for this reason it is not depicted.

- The RTDs do not have all the same shape. Cross-overs occur between algorithms in the medium size instances.

- In the medium size test set the median cost value of the hardest instance is 525 times the median cost value of the easiest for RR.LS, more than 11 times for SA, ILS and ACO, and almost the same for SA.LS. In the small size test set, instead, the factor is 7 in the case of SA and the same for ILS and ACO.

- SA.LS shows robustness w.r.t. all kind of instances. Indeed it seems that the harder the instance the better it performs compared to the others algorithms. Remarkable is, for instance, what happens on the hardest instance of the small size test set. There, SA, which seemed to dominate SA.LS, shows a behavior which is significatively less robust than SA.LS. Moreover, on the hardest instance of the medium size test-set SA shows the risk of not finding a feasible solution even after 1800 seconds.

- For SA on the hardest instances of both small and medium size set after a certain time t_0 the distributions become suddenly much less steep. This indicates that after t_0 the

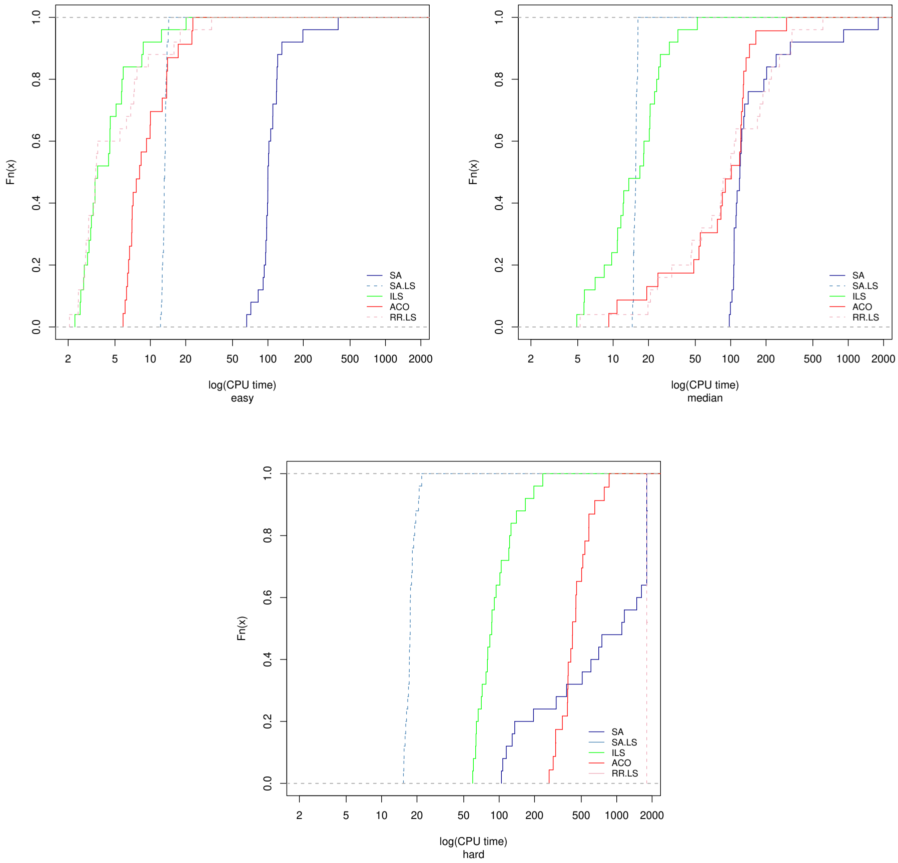


Figure 4: RTDs of algorithms on the easiest (easy), the median (median) and the hardest (hard) instances of the *medium* size test set, based on 25 tries per instance. The x axis gives the logarithm of the CPU time, the y axis the empirical cumulative distribution function $Fn(x)$. Note that RR.LS is never able to solve the hardest instance in 1800 seconds.

behavior of the algorithms gets worse because the probability of finding a feasible solution increases less than before t_0. This phenomenon studied already in the case of the TSP and the SAT problem [13, 7, 4] is recognized as *stagnation* behavior and can be avoided by restarting the algorithm at some appropriately chosen time.

- The absolute distance between the minimum and the maximum time needed for finding a feasible solution seems not to vary considerably for ACO and ILS in the respective size test-set while for SA its evident that it increases with the hardness of the instance.

3 Summarizing RTDs with continuous probability distributions

Trying to fit the empirical cumulative distribution of the run time with a continuous distribution can lead to a better understanding of the experimental results (mean, median, quantiles) [2].

Related to the distribution fitting is also the study about the robustness w.r.t. the cutoff

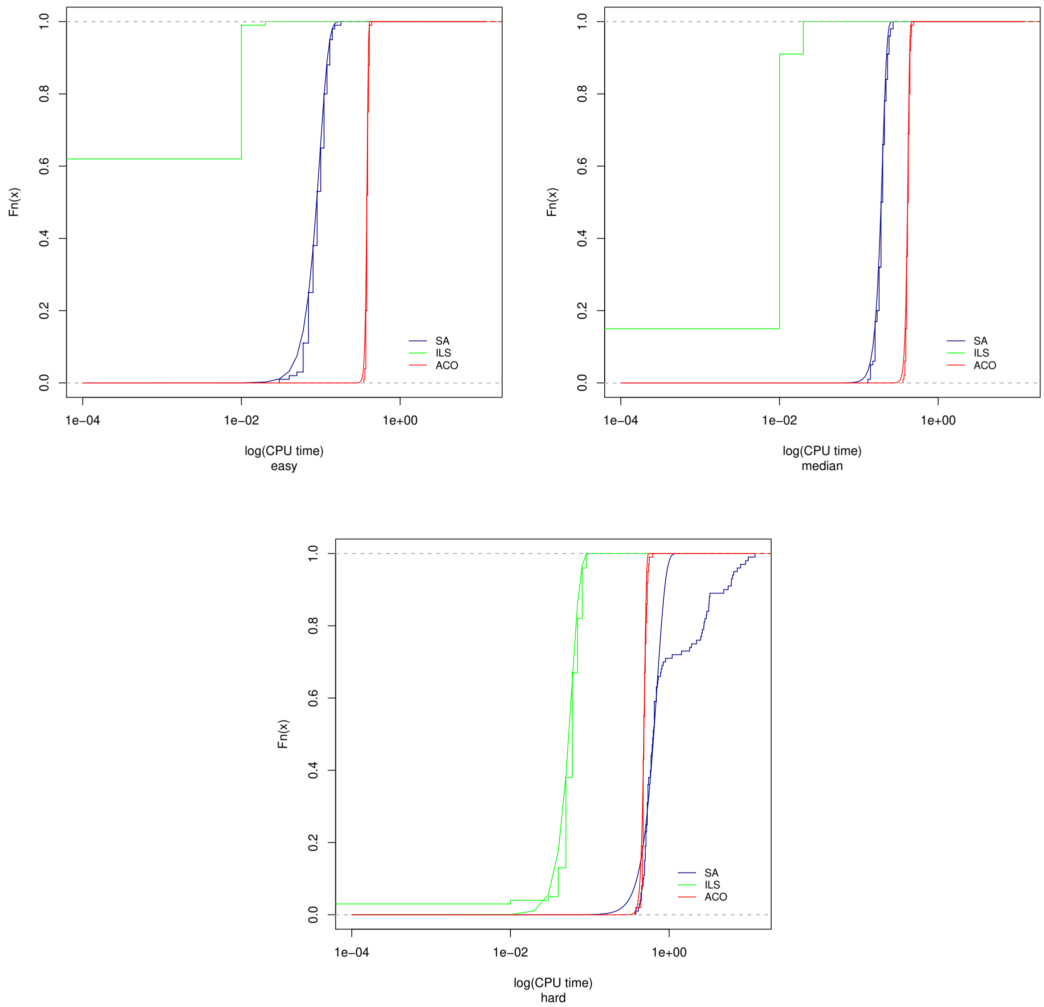


Figure 5: RTDs of algorithms fitted by Weibull distributions on the easiest (easy), the median (median) and the hardest (hard) instance of the *small* size test set, based on 100 trials per instance. The x axis gives the logarithm of the CPU time, the y axis the empirical cumulative distribution function $Fn(x)$. In order to avoid confusion in the graphs RR.LS and SA.LS are not depicted since their behavior is similar to ILS and ACO, respectively.

parameters. If stagnation occurs, the maximal time allowed for the search (cutoff time) is too big and a simple restart of the search from a new random initial solution can help in finding more quickly solutions [13]. The maximal robustness w.r.t. the cutoff parameter is encountered if the RTD is an exponential distribution. This is because an exponential RTD is memoryless in the sense that the probability of finding a solution within a fixed time does not depend on the time passed so far. Consequently, as already observed for TSP and SAT analysis [5, 13, 7, 4], in case of an exponential RTD, the probability to find a solution within a given time would not depend on the number of random restarts, while when the experimental distribution is not fitted by an exponential distribution it may be possible to find a point in the time axis for which a restart can be fruitful.

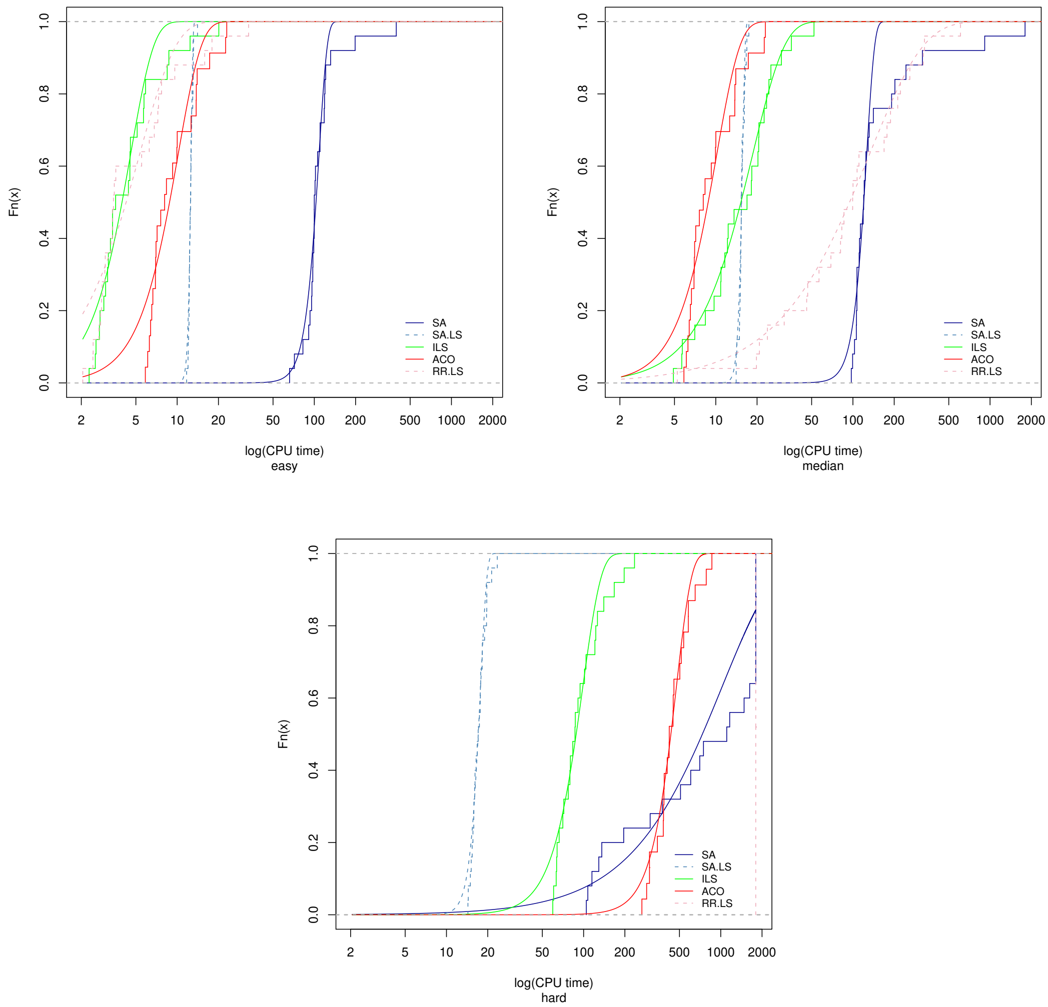


Figure 6: RTDs of algorithms fitted by Weibull distributions on the easiest (easy), the median (median) and the hardest (hard) instance of the *medium* size test set, based on 25 trials per instance. The x axis gives the logarithm of the CPU time, the y axis the empirical cumulative distribution function $Fn(x)$.

In our case, for each individual instance and algorithm, the RTD is not fitted properly by an exponential distribution but it is better approximated by the Weibull cumulative distribution $F(t) = 1 - e^{-(t/b)^a}$ where t is the time [7]. The approximation of the parameters a and b for fitting the experimental distribution is done with the least-squares non-linear regression under the statistical environment R. In all cases the fit between RTDs and ideal Weibull distributions passed the Kolmogorov Smirnov test [12].

The Weibull distribution has not the same memoryless properties of the exponential one so the effect of a random restart cannot easily be judged. In some cases, however, it is possible to shift an exponential distribution to fit at least an initial part of RTD, as done in [13]. This can be done only with SA in the hard instances of Figure 5 and Figure 6 and it indicates that SA could gain from occasional random restarts.

In our case it is impossible, at least for the moment being, to assert *a priori* that an instance is harder than another one, thus looking for an optimal instance dependent cutoff point is useless. But the information that a restart may improve performance is useful if we define restart criteria based on the measurement of the number of iterations since the last improvement of the evaluation function value. In this sense an algorithm could be restarted when for a fixed number of iterations it does not produce any improved solution.

Nevertheless knowing that our experimental distributions are well fitted by a Weibull distribution can be relevant for others benefits like the possibility to estimate results from only few data or the aid in the interpretation of experiments in which runs are terminated prematurely after a certain point.

4 Scaling with instance hardness

When we compared the hardness distributions of the median run time cost over the set of instances in Figure 2, we observed that these hardness distributions are mostly similarly shaped and with prominent tails. As a consequence of this and as suggested in [7, 6], the hardness of instances for different algorithms might be correlated.

For testing this, in Figure 7 we plot, for each pair of algorithms, the median search cost w.r.t. the same instance and we compute the correlation. We can conclude that:

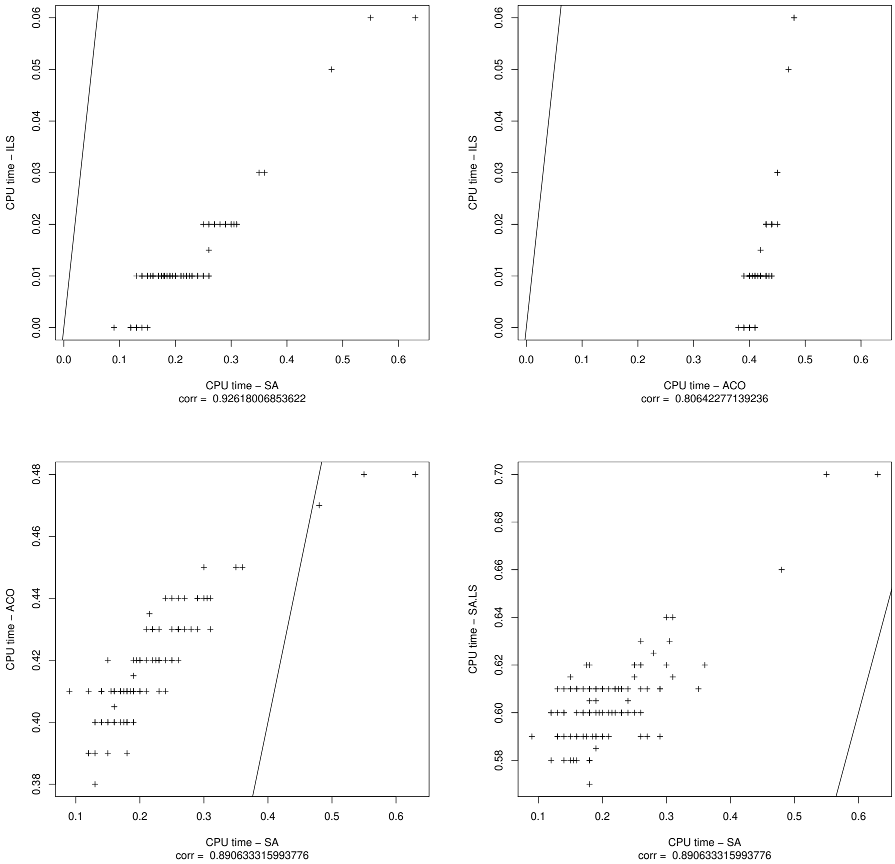
- Both, Figure 7 and Figure 8, confirm the hypothesis that there is a tight correlation between the performance of any two algorithm over the test-set. We conclude that instances which are hard for one algorithm tend to be hard also for the other algorithms.
- SA.LS shows less correlation w.r.t. the other algorithms. This confirms, as noted before, its robustness w.r.t. instance hardness and suggests a possible use of this algorithm for hybrid combinations.
- The hardest instance found is the same for all algorithms and for this instance RR.LS is never able to find a feasible solution in 1800 seconds.

5 Scaling with instance size

Many parameters influence the way an instance is built. Here we consider the instance generator as a black–box and we investigate how the different parameters for the instance generation influence the search cost of the algorithms. More specifically, we analyze independently the behavior w.r.t. the maximum number of students per event, the number of rooms available, the number of features and the approximate number of features per room. The last two parameters are somehow correlated, so we consider a ratio between them based on different values of the number of features. We keep fixed, instead, the number of events, the number of students and the maximum number of events per students, the first two because they are already considered as the main parameters influencing the size and the latter one because a higher number of events a student should attend in a week seems quite improbable. This analysis is done for the small sized instances because of the computation time, but we believe that the results can be extended to the medium and large size instances. The median and the 90% quantile of 100 runs per instance are reported. ACO is not represented since, as observed before, the computation time spent by ACO for solving the hard constraints does not vary very much.

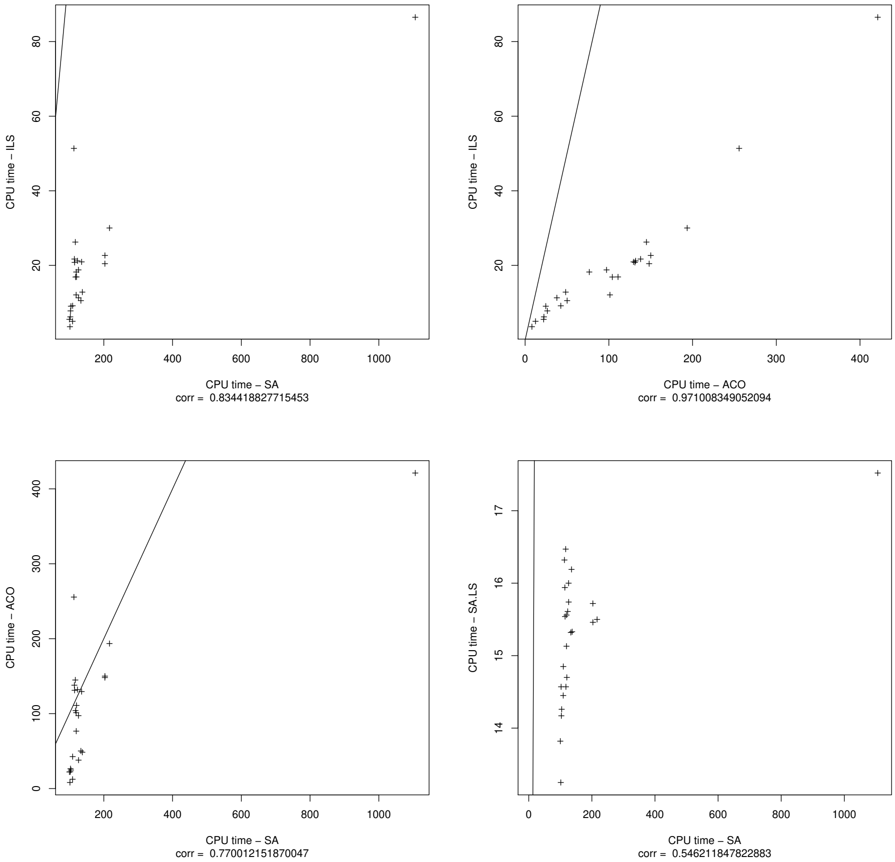
The following conclusions can be drawn:

- The data shows that the CPU time increases with the number of students. The points in the graphs are too few for quantifying correctly this growth but it seems, remembering



correlation	SA	SA.LS	ILS	ACO
SA.LS	0.7840797	–	–	–
ILS	0.9261801	0.7674426	–	–
ACO	0.8906333	0.6005224	0.8064228	–
RR.LS	0.9161625	0.7821565	0.9422633	0.7779474

Figure 7: Correlation between pairs of algorithms on the *small* size test-set. Each single point gives the median on 100 runs of the two algorithms considered. For example, in the comparison between ILS and SA the point in the top right corner of the graphs indicates that for solving that specific instance ILS requires a median value of 0.06 seconds of CPU time while SA requires 0.63 seconds of CPU time. The line indicates points of equal CPU time for the two algorithms. The table summarizes the correlation coefficient for algorithm pairs included those which are not depicted.



correlation	SA	SA.LS	ILS	ACO
SA.LS	0.5462118	–	–	–
ILS	0.8344188	0.7777637	–	–
ACO	0.7700122	0.7944625	0.9710083	–
RR.LS	0.5996755	0.5683776	0.8357960	0.8424872

Figure 8: Correlation between pairs of algorithms on the *medium* size test-set. Each single point gives the median on 25 runs of the two algorithms considered. The line indicates points of equal CPU time for the two algorithms. The point on the top right corner of each graph represent the instance which resulted to be the hardest for all the algorithms (it was number "11" generated by random seed equal to 1100). The table summarizes the correlation coefficient for algorithm pairs included those which are not depicted.

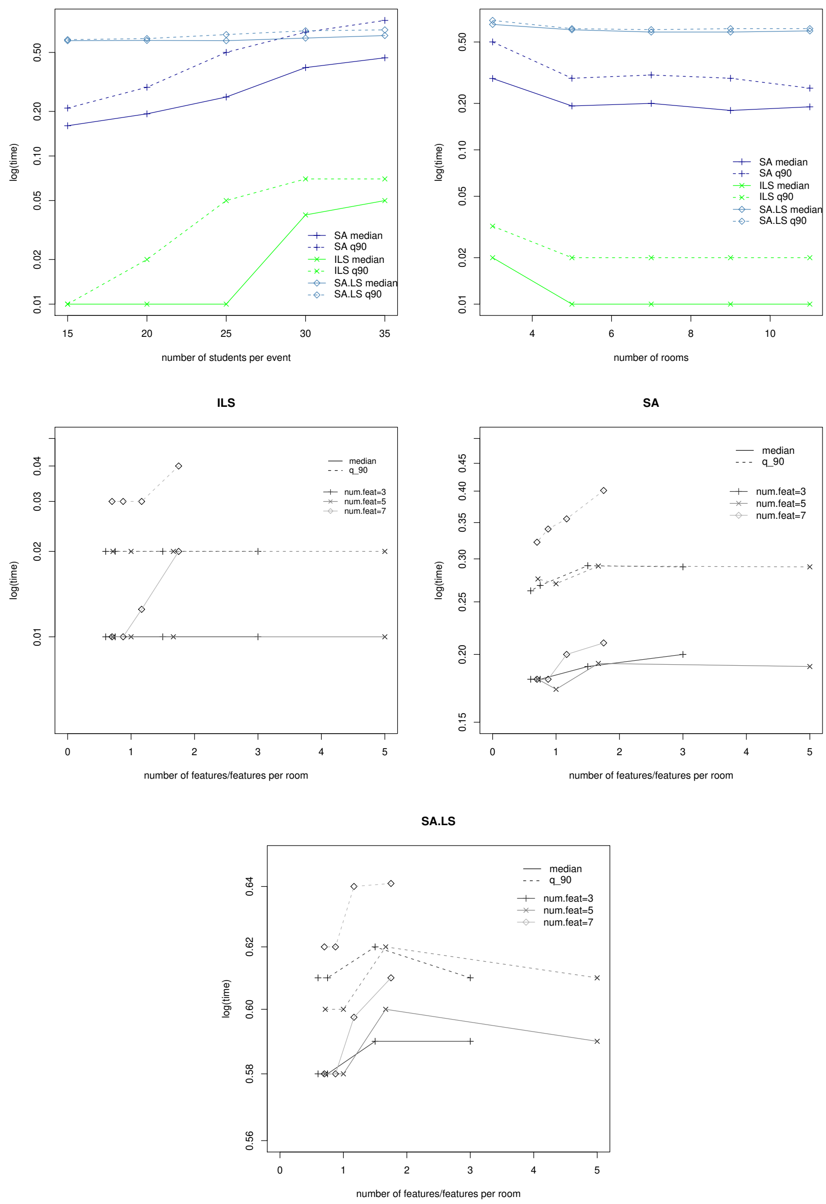


Figure 9: Scaling of algorithms with respect to instance generator parameters. The two upper graphs represent results for SA, ILS and SA.LS and the parameter that is changed in the instance generator is indicated on the abscissa. The two lower graphs represent respectively SA, ILS, and SA.LS results where in the abscissa axis is indicated the ratio between the number of features and approximate features per room and the different values assumed for the number of features is indicated in the legend. Only small size instances are considered. The Graphs are semilogarithmic.

that the graph is logarithmic, that the effects of the growth are more strong for SA than for ILS. That is, if the behavior is linearly the steep of SA is higher than that of ILS.

- The behavior of SA, SA.LS and ILS seems decreasing w.r.t. the number of rooms. This is reasonable since the larger the number of rooms available the easier it should be the room assignment by the matching algorithm. Nevertheless after the value 5 as number of rooms the computation time seems not to vary significatively.

- For the other two parameters, number of features, and approximate number of features per room, it seems there is a value for the ratio which produces instances harder than others. Before this value the solution times increase and after that they are decreasing again.

6 Conclusions

We showed that there can be significant difference in the instance hardness. This effect becomes more prominent as the instance size grows and can become extremely evident, as observed in the case of SA, ACO and ILS on the medium size test-set where a feasible solution can be found in a range of time larger than 100 seconds (figure 2). We observed that the hardness of an instance appears to be an instance intrinsic property. This was confirmed by the high correlation between the computation time needed by pairs of algorithms when solving the same instance.

We showed how the distributions of the times for solving an instance are quite similar to the Weibull continuous distribution. From this many advantages can arise, as the possibility to estimate the shape and scale parameters from only a small numbers of experiments, or the possibility to convey a complete understanding of the experimental results which can be often, like in the case of the mean, the median, the mode, or the quantile, only partial representation of the underlying distributions.

We, then, studied the robustness of algorithms related to the maximal time allowed for the search and we discovered that for SA on the hard instances, a simple restart from a new random initial solution after a certain time of the search could improve the performance. Due to the impossibility of knowing *a priori* the hardness of an instance, restart could occur after a certain number of iterations without improvement of the best solution found so far.

Concerning the instances, we gave indication about how they are affected by the parameters required by the instance generator. In particular, hardness increases by increasing the number of students per event or by specific combinations of the number of features and approximate features per rooms while it decreases by increasing the number of rooms.

Finally, the analysis performed provided also a clearer insight into the performance of the algorithms and new results. Considering only the hard constraints, indeed, we discovered that the algorithms which seem to perform best are SA.LS and ILS. This is a relevant result since in the automated tuning used at Intellectics Groups [1] SA performed better than SA.LS when all kind of constraints (hard and soft) are considered. Therefore it seems that SA.LS works better (*i.e.* it is more robust) with respect to only the hard constraints and a hybrid version which uses SA.LS for solving hard constraints and SA for solving the soft one should improve significantly the performance of Simulated Annealing.

Acknowledgments

We would like to thanks Michael Samples, Mauro Birattari and Luis Paquete for discussions and suggestions on the topic of this report.

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] Mauro Birrattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. Technical Report AIDA–02–01, FG Intellektik, TU Darmstadt, January 2002.

[2] Daniel Frost, Irina Rish, and Lluis Vila. Summarizing CSP hardness with continuous probability distributions. In *Proceedings of AAAI*, 1997.

[3] T. Hogg. Refining the phase transition in combinatorial search. *Artificial Intelligence*, 81(1-2):127–154, 1996.

[4] Holger H. Hoos and Thomas Stützle. Evaluating Las Vegas algorithms – pitfalls and remedies. In *UAI-98*, pages 238–245. Morgan Kaufmann, 1998.

[5] Holger H. Hoos and Thomas Stützle. Systematic vs. local search for SAT. In Wolfram Burgard, Thomas Christaller, and Armin B. Cremers, editors, *KI-99: Advances in Artificial Intelligence*, volume 1701 of *Lecture Notes in Artificial Intelligence*, pages 289–293, Berlin, 1999. Springer Verlag.

[6] Holger H. Hoos and Thomas Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence*, 112(1–2):213–232, 1999.

[7] Holger H. Hoos and Thomas Stützle. Local search algorithms for SAT: An empirical evaluation. In Ian Gent, Hans van Maaren, and Toby Walsh, editors, *SAT2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*, pages 43–88. IOS Press, 2000.

[8] Olivia Rossi-Doria and Ben Paechter. A local search for the timetabling problem. To appear in PATAT 2002: The 4th international conference on the Practice And Theory of Automated Timetabling. Gent, Belgium, August 21-23, 2002.

[9] Olivia Rossi-Doria, Michael Samples, Marco Chiarandini, Joshua Knowles, Max Manfrin, Monaldo Mastrolilli, Luis Paquete, and Ben Paechter. A comparison of the performance of different metaheuristics on the timetabling problem. To appear in PATAT 2002: The 4th international conference on the Practice And Theory of Automated Timetabling. Gent, Belgium, August 21-23, 2002.

[10] B. Selman and S. Kirkpatrick. Critical behavior in the computational cost of satisfiability testing. *Artificial Intelligence*, 81(1-2):273–295, 1996.

[11] Bart Selman, David G. Mitchell, and Hector J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1-2):17–29, 1996.

[12] Sidney Siegel and Jr. N. John Castellan. *Nonparametric system for the behavioral sciences.* McGraw-Hill, second edition, 1988.

[13] T. Stützle and H. H. Hoos. Analyzing the run-time behaviour of iterated local search for the TSP. In P. Hansen and C. Ribeiro, editors, *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, 2001.