Consultant-Guided Search Algorithms with Local Search for the Traveling Salesman Problem

Serban Iordache

SCOOP Software GmbH, Köln, Germany
siordache@acm.org

Abstract. Consultant-Guided Search (CGS) is a recent metaheuristic for combinatorial optimization problems, which has been successfully applied to the Traveling Salesman Problem (TSP). In experiments without local search, it has been able to outperform some of the best Ant Colony Optimization (ACO) algorithms. However, local search is an important part of any ACO algorithm and a comparison without local search can be misleading. In this paper, we investigate if CGS is still able to compete with ACO when all algorithms are combined with local search. In addition, we propose a new variant of CGS for the TSP, which introduces the concept of confidence in relation to the recommendations made by consultants. Our experimental results show that the solution quality obtained by this new CGS algorithm is comparable with or better than that obtained by Ant Colony System and MAX-MIN Ant System with 3-opt local search.

Keywords: Metaheuristics, combinatorial optimization, swarm intelligence, traveling salesman problem.

1 Introduction

Consultant-Guided Search (CGS) [5] is a swarm intelligence technique based on the direct exchange of information between individuals in a population. It takes inspiration from the way real people make decisions based on advice received from consultants. CGS can be used to solve hard combinatorial optimization problems and it has been first applied to the Traveling Salesman Problem (TSP). Experimental results have shown that the CGS-TSP algorithm introduced in [5] can outperform some of the best Ant Colony Optimization (ACO) [3] algorithms. However, these results may be misleading, because the experiments have been performed without local search and, in practice, ACO algorithms for the TSP are always combined with local search. While most metaheuristics work better when combined with local search, the performance improvement can vary significantly from one algorithm to another. Therefore, the main goal of this paper is to investigate whether CGS is still able to compete with ACO when the algorithms are combined with local search.

Moreover, we propose a variant of the CGS-TSP algorithm, where consultants also indicate the confidence in the recommendations they make. In experiments

R. Schaefer et al. (Eds.): PPSN XI, Part II, LNCS 6239, pp. 81–90, 2010.

with 3-opt local search, we compare the performance of this new algorithm to that of the standard CGS-TSP algorithm, as well as to that of Ant Colony System (ACS) [2] and MAX-MIN Ant System (MMAS) [7].

This paper is organized as follows: to make the paper self-contained, we describe in Section 2 the CGS metaheuristic and the standard CGS-TSP algorithm; in Section 3, we combine CGS with local search, we introduce the new CGS-TSP variant that attaches confidence levels to recommendations and we report our experimental results; in Section 4 we conclude the paper and present future research directions.

2 The CGS Metaheuristic and Its Application to the TSP

In this section, we briefly describe the CGS metaheuristic and the CGS-TSP algorithm. We refer the reader to [5] for a more detailed presentation.

2.1 The Metaheuristic

CGS is a population-based method. An individual of the CGS population is a virtual person, which can simultaneously act both as a client and as a consultant. As a client, a virtual person constructs at each iteration a solution to the problem. As a consultant, a virtual person provides advice to clients, in accordance with its ***strategy***. Usually, at each step of the solution construction, there are several variants a client can choose from. The variant recommended by the consultant has a higher probability to be chosen, but the client may opt for one of the other variants, which will be selected based on some heuristic.

At the beginning of each iteration, a client chooses a consultant based on its ***personal preference*** and on the consultant's ***reputation***. The reputation of a consultant increases with the number of successes achieved by its clients. A client achieves a ***success***, if it constructs a solution better than all solutions found until that point by any client guided by the same consultant. Each time a client achieves a success, the consultant adjusts its strategy in order to reflect the sequence of decisions taken by the client. Because the reputation fades over time, a consultant needs that its clients constantly achieve successes, in order to keep its reputation. If the consultant's reputation sinks below a minimum value, it will take a ***sabbatical leave***, during which it will stop offering advice to clients and it will instead start searching for a new strategy to use in the future.

The pseudocode that formalizes the CGS metaheuristic is shown in Fig. 1. During the initialization phase (lines 2-5), virtual people are created and placed in sabbatical mode. Based on its mode, a virtual person constructs at each iteration either a solution to the problem (line 13) or a consultant strategy (line 9). A local optimization procedure (line 17) may be applied to improve this solution or consultant strategy.

After the construction phase, a virtual person in sabbatical mode checks if it has found a new best-so-far strategy (lines 20-22), while a virtual person in

```
 1. procedure CGSMetaheuristic()
 2.     create the set P of virtual persons
 3.     foreach p ∈ P do
 4.       setSabbaticalMode(p)
 5.     end foreach
 6.     while(termination condition not met) do
 7.       foreach p ∈ P do
 8.         if actionMode[p]=sabbatical then
 9.           currStrategy[p] ← constructStrategy(p)
10.         else
11.           currCons[p] ← chooseConsultant(p)
12.           if currCons[p] ≠ null then
13.             currSol[p] ← constructSolution(p, currCons[p])
14.           end if
15.         end if
16.       end foreach
17.       applyLocalOptimization()                    // optional
18.       foreach p ∈ P do
19.         if actionMode[p]=sabbatical then
20.           if currStrategy[p] better than bestStrategy[p] then
21.             bestStrategy[p] ← currStrategy[p]
22.           end if
23.         else
24.           c ← currCons[p]
25.           if c ≠ null and currSol[p] is better than all solutions
26.                         found by a client of c since last sabbatical then
27.             successCount[c] ← successCount[c]+1
28.             strategy[c] ← adjustStrategy(c, currSol[p])
29.           end if
30.         end if
31.       end foreach
32.       updateReputations()
33.       updateActionModes()
34.     end while
35. end procedure
```

Fig. 1. The CGS Metaheuristic

normal mode checks if it has achieved a success and, if this is the case, it adjusts its strategy accordingly (lines 24-29).

Reputations are updated based on the results obtained by clients (line 32): the reputation of a consultant is incremented each time one of its clients achieves a success and it receives an additional bonus when a client obtains a best-so-far result. Each consultant is ranked based on the best result obtained by any client working under its guidance. For a number of top-ranked consultants, CGS prevents their reputations from sinking below a predefined level.

Finally, the action mode of each virtual person is updated (line 33): consultants whose reputations have sunk below the minimum level are placed in

sabbatical mode, while consultants whose sabbatical leave has finished are placed in normal mode.

2.2 The CGS-TSP Algorithm

The CGS-TSP algorithm introduced in [5] is an application of the CGS metaheuristic to the TSP. In this algorithm, the ***strategy*** of a consultant is represented by a tour, which it advertises to its clients. The heuristic used to build new strategies during the ***sabbatical leave*** is based on a pseudorandom proportional rule that strongly favors the nearest city. A virtual person k located at city i moves to a city j according to the following rule:

$$j = \begin{cases} argmin_{l \in \mathcal{N}_i^k}\{d_{il}\} & \text{if } a \leq a_0, \\ J & \text{otherwise.} \end{cases} \tag{1}$$

where: $\mathcal{N}_i^k$ is the feasible neighborhood of person k when being at city i; d_{il} is the distance between cities i and l; a is a random variable uniformly distributed in $[0, 1]$ and $a_0 \in [0, 1]$ is a parameter; J is a random variable selected according to the probability distribution given by formula (2), where β is a parameter.

$$p_{ij}^k = \frac{(1/d_{ij})^\beta}{\sum_{l \in \mathcal{N}_i^k}(1/d_{il})^\beta} \tag{2}$$

Based on the tour advertised by the consultant, a client receives at each step of the solution construction a recommendation regarding the next city to be visited. The client does not always follow the consultant's recommendation. Again, a pseudorandom proportional rule is used to decide which city to visit at the next step:

$$j = \begin{cases} v & \text{if } v \neq null \wedge q \leq q_0, \\ argmin_{l \in \mathcal{N}_i^k}\{d_{il}\} & \text{if } (v = null \vee q > q_0) \wedge b \leq b_0, \\ J & \text{otherwise.} \end{cases} \tag{3}$$

where: v is the city recommended by the consultant for the next step; q is a random variable uniformly distributed in $[0, 1]$ and q_0 ($0 \leq q_0 \leq 1$) is a parameter; $\mathcal{N}_i^k$ is the feasible neighborhood of person k when being at city i; d_{il} is the distance between cities i and l; b is a random variable uniformly distributed in $[0, 1]$ and b_0 is a parameter ($0 \leq b_0 \leq 1$); J is a random variable selected according to the probability distribution given by formula (2).

In CGS-TSP, the ***personal preference*** is given by the inverse of the advertised tour length. It is used by clients in conjunction with the ***reputation***, in order to compute the probability to choose a given consultant k:

$$p_k = \frac{reputation_k^\alpha \, preference_k^\gamma}{\sum_{c \in \mathcal{C}}(reputation_c^\alpha \, preference_c^\gamma)} \tag{4}$$

where $\mathcal{C}$ is the set of all available consultants and α and γ are parameters that determine the influence of reputation and personal preference.

At each iteration, the consultant's k reputation fades at rate r:

$$reputation_k \leftarrow reputation_k(1 - r) \tag{5}$$

CGS-TSP adjusts its reputation fading rate according to the total number s_w of successes achieved during the last w iterations by the best $fadingRanks$ consultants, where w and $fadingRanks$ are parameters:

$$r = r_0 \left(1 + \frac{s_w}{\sqrt{1 + \left(\frac{s_w}{f}\right)^2}} \right) \tag{6}$$

The parameter r_0 gives the reputation fading rate for the case where no successes were achieved by the best $fadingRanks$ consultants during the last w iterations. The value of f is computed from the value of another parameter, k_w, which indicates how much greater is the decrease in reputation for a very high number of successes than the decrease in the case when no successes were achieved:

$$f = \left(\frac{1}{r_0} - 1\right)\left(1 - \frac{1}{\sqrt[w]{k_w}}\right) \tag{7}$$

3 CGS with Local Search for the TSP

In this section, we present the details of combining CGS-TSP with local search and we propose a new variant of this algorithm, which introduces the concept of confidence in relation to the recommendations made by consultants. Then, we describe the experimental setting and we compare the results obtained by the competing algorithms considered.

3.1 Applying Local Search to CGS-TSP

Since the CGS metaheuristic provides an optional local optimization step, combining CGS-TSP with local search is a straightforward process. Local search improves the results, but it is a time-consuming procedure. Therefore, significantly fewer iterations can be performed in the same amount of time when the algorithm is combined with local search. For this reason, different parameter settings are appropriate in this case. The standard CGS-TSP algorithm fixes the value of the parameter w to 1000 and the $sabbaticalDuration$ to 100 iterations. In our experiments with local search we fix the value of w to 100 and the $sabbaticalDuration$ to $500/m$ iterations, where m is the number of virtual persons. For the other parameters with fixed values in the standard CGS-TSP, we preserve the original values when combining the algorithm with local search: $a_0 = 0.9$ and $minReputation = 1$.

3.2 CGS-TSP with Confidence

In addition to the algorithm described in the previous subsection, we propose a variant of the CGS-TSP algorithm, which we refer to as CGS-TSP-C, where each arc in the tour advertised by a consultant has an associated ***strength***. Strengths are updated each time the consultant adjusts its strategy. If an arc in the new advertised tour was also present in the old advertised tour, its strength will be incremented; otherwise, its strength is set to 0. The strength of an arc could be interpreted as the consultant's confidence in recommending this arc to a client. A client is more likely to accept recommendations made with greater confidence. This idea is expressed in CGS-TSP-C by allowing the value of the parameter q_0 to vary in a given range, at each construction step:

$$q_0 = \begin{cases} q_{min} + s \cdot \frac{q_{max} - q_{min}}{s_{max}} & \text{if } s < s_{max}, \\ q_{max} & \text{otherwise.} \end{cases} \tag{8}$$

where s is the strength of the recommended arc and q_{min}, q_{max} and s_{max} are parameters.

3.3 Experimental Setup

To allow a meaningful comparison between heuristics, we have created a software package containing Java implementations of the algorithms considered in our experiments. The software package is available as an open source project at `http://swarmtsp.sourceforge.net/`. At this address, we also provide all configuration files, problem instances and results files for the parameter tuning and for the experiments described in this paper, as well as the outcome of other experiments briefly mentioned in this paper, but not further detailed due to space limitations.

We run a series of experiments in order to compare the performance of CGS-TSP and CGS-TSP-C with that of Ant Colony System (ACS) [2] and MAX-MIN Ant System (MMAS) [7]. We combine all algorithms used in our experiments with 3-opt local search and we use candidate lists of length 20 for all algorithms. Each run is terminated after $n/50$ seconds CPU time, where n is the problem size.

We have tuned the parameters of all algorithms through the ParamILS [4] and F-Race [1] procedures. As training set, we have used 600 generated Euclidean TSP instances, with the number of cities uniformly distributed in the interval $[1000, 2000]$. For CGS-TSP and CGS-TSP-C, the parameter settings are given in Table 1.

The best configuration found for ACS is: $m = 12$, $\rho = 0.6$, $\xi = 0.4$, $q_0 = 0.98$, $\beta = 2$. For MMAS, the best configuration found is: $m = max(10, 44 - 0.175 \cdot n)$, $\rho = 0.15$, $\alpha = 2$, $\beta = 2$. Some of these values differ significantly from the standard values given in [3, p.96], which are: $m = 10$, $\rho = 0.1$, $\xi = 0.1$, $q_0 = 0.98$, $\beta = 2$ for ACS and: $m = 25$, $\rho = 0.2$, $\alpha = 1$, $\beta = 2$ for MMAS. For this reason, for ACS and MMAS, we have performed our experiments using both the tuned and

Table 1. Parameter settings for CGS-TSP and CGS-TSP-C

Parameter	CGS-TSP	CGS-TSP-C	Description
m	$max(3, 21 - n/125)$	$max(3, 16 - n/250)$	number of virtual persons
b_0	0.98	0.95	*see formula* (3)
q_0	0.98	$\begin{cases} q_{min} = 0.8 \\ q_{max} = 0.99 \\ s_{max} = 3 \end{cases}$	*see formulas* (3), (8)
α	7	7	reputation's relative influence
β	12	12	heuristic's relative influence
γ	7	8	result's relative influence
$maxReputation$	40	50	maximum reputation value
$initialReputation$	6	3	reputation after sabbatical
$bonus$	8	6	best-so-far reputation bonus
$protectedRanks$	$0.7 \cdot m$	$0.6 \cdot m$	protected top consultants
r_0	$3 \cdot 10^{-7}$	$3 \cdot 10^{-6}$	basic reputation fading rate
$fadingRanks$	2	10	top consultants for fading rate
k_w	3	30	reputation decrease factor

the standard values. As shown in the next subsection, the results obtained using the tuned parameters outperform those obtained with the standard settings.

3.4 Experimental Results

We have applied the algorithms to 27 symmetric instances from the TSPLIB benchmark library, with the number of cities between 654 and 3038. Table 2 reports for each algorithm and TSP instance the best and mean percentage deviations from the optimal solutions over 25 trials. The best mean results for each problem are in boldface. We also report for each problem the p-values of the one-sided Wilcoxon rank sum tests for the null hypothesis (H_0) that there is no difference between the solution quality of CGS-TSP-C and that of the competing ACO algorithm, and for the alternative hypothesis (H_1) that CGS-TSP-C outperforms the considered algorithm. Applying the Bonferroni correction for multiple comparisons, we obtain the adjusted α-level: $0.05/27 = 0.00185$. The p-values in boldface indicate the cases where the null hypothesis is rejected at this significance level.

For a few pairs of algorithms, we use the one-sided Wilcoxon signed rank test to compute the p-values for the null hypothesis (H_0) that there is no difference between the means of the first and the means of the second algorithm considered, and the alternative hypothesis (H_1) that the means of the first algorithm are smaller than the means of the second algorithm considered. The p-values are given in Table 3.

The null hypothesis can be rejected at a high significance level when CGS-TSP-C is compared with the two ACO algorithms, which means that for runs terminated after n/50 seconds CPU time, CGS-TSP-C clearly outperforms both

Table 2. Performance over 25 trials. Runs are terminated after $n/50$ CPU seconds.

Problem instance	ACS		MMAS		CGS-TSP		CGS-TSP-C			
	Best (%)	Mean (%)	Best (%)	Mean (%)	Best (%)	Mean (%)	Best (%)	Mean (%)	p-value (ACS)	p-value (MMAS)
p654	0.000	0.023	0.000	0.062	0.000	**0.009**	0.000	**0.009**	**0.0004**	**0.0000**
d657	0.002	0.167	0.031	0.133	0.002	0.129	0.002	**0.097**	0.0029	0.0326
gr666	0.056	0.159	0.000	**0.066**	0.040	0.069	0.000	0.109	0.0142	0.8059
u724	0.026	0.102	0.045	0.151	0.005	0.128	0.005	**0.101**	0.5932	0.0070
rat783	0.000	0.252	0.023	0.185	0.000	0.215	0.000	**0.147**	0.0157	0.1564
dsj1000	0.038	0.403	0.133	0.316	0.030	0.296	0.000	**0.224**	0.0042	0.1315
pr1002	0.000	0.273	0.034	0.201	0.000	0.189	0.000	**0.162**	0.0203	0.0863
si1032	0.000	**0.023**	0.000	0.035	0.000	0.029	0.000	0.024	0.6031	0.1758
u1060	0.116	0.331	0.104	0.359	0.026	**0.117**	0.026	0.119	**0.0000**	**0.0000**
vm1084	0.000	**0.064**	0.001	0.120	0.000	0.066	0.000	0.071	0.5096	**0.0012**
pcb1173	0.002	0.325	0.021	**0.219**	0.185	0.449	0.002	0.297	0.2456	0.9376
d1291	0.000	0.111	0.000	**0.105**	0.000	0.108	0.000	0.186	0.3690	0.4518
rl1304	0.000	0.196	0.000	0.229	0.000	0.200	0.000	**0.189**	0.5291	0.0243
rl1323	0.077	0.243	0.041	0.245	0.000	**0.131**	0.010	0.152	**0.0005**	**0.0000**
nrw1379	0.088	**0.256**	0.305	0.486	0.083	0.286	0.152	0.264	0.5668	**0.0000**
fl1400	0.020	0.247	0.298	0.668	0.000	0.190	0.000	**0.177**	0.0046	**0.0000**
u1432	0.218	**0.389**	0.250	0.601	0.292	0.481	0.153	0.426	0.8654	**0.0000**
fl1577	0.031	0.293	0.220	0.597	0.004	**0.060**	0.004	0.172	**0.0001**	**0.0000**
d1655	0.006	0.435	0.019	0.325	0.064	0.332	0.000	**0.322**	0.0754	0.4446
vm1748	0.113	0.272	0.154	0.471	0.061	0.191	0.004	**0.159**	**0.0000**	**0.0000**
u1817	0.192	0.480	0.080	**0.295**	0.156	0.386	0.107	0.347	0.0043	0.9048
rl1889	0.345	0.719	0.270	0.545	0.004	0.237	0.000	**0.217**	**0.0000**	**0.0000**
d2103	0.017	0.332	0.040	0.127	0.000	0.345	0.000	**0.047**	**0.0000**	**0.0000**
u2152	0.112	0.426	0.254	0.488	0.115	0.356	0.131	**0.352**	0.0472	**0.0015**
u2319	0.287	**0.372**	0.427	0.599	0.707	0.863	0.742	1.052	1.0000	1.0000
pr2392	0.235	0.507	0.214	0.542	0.019	0.441	0.051	**0.340**	**0.0001**	**0.0012**
pcb3038	0.243	**0.499**	0.671	0.940	0.704	1.056	0.428	0.810	1.0000	**0.0003**

ACS and MMAS. In addition, CGS TSP C outperforms CGS-TSP, which means that the use of confidence in relation to the recommendations made by consultants can lead to better results.

As shown in [5], CGS-TSP clearly outperforms ACS and MMAS in experiments without local search. Combined with 3-opt local search, CGS-TSP still outperforms ACS and MMAS, but only at a moderate significance level. This means that ACS and MMAS benefit more than CGS-TSP from the hybridization with local search. One explanation for this discrepancy could be that the solution construction mechanism of CGS-TSP already bears some resemblance to a local search procedure: a client builds a solution in the neighborhood of the solution promoted by a consultant, while the consultant updates its promoted solution each time one of its clients finds a better one.

Table 3. Performance comparison using the one-sided Wilcoxon signed rank test

First algorithm	Second algorithm	p-value
CGS-TSP-C	ACS	0.00472
CGS-TSP-C	MMAS	0.00148
CGS-TSP-C	CGS-TSP	0.02004
CGS-TSP (without local search)	ACS (without local search)	$*$ 0.00003
CGS-TSP	ACS	0.05020
CGS-TSP (without local search)	MMAS (without local search)	$*$ 0.00269
CGS-TSP	MMAS	0.03051
ACS	ACS (standard settings)	0.00256
MMAS	MMAS (standard settings)	$<$ 0.00001

$*$ p-values taken from [5].

As mentioned in the previous subsection, some of the parameter values found during the tuning phase for ACS and MMAS differ significantly from the values recommended in [3, p.96]. The last two lines of Tab. 3 compare the performance obtained using the tuned parameters and the standard settings. The tuned algorithms clearly outperform the algorithms that use standard settings, thus confirming the effectiveness of the tuning procedures.

Figure 2 shows the development of the mean percentage deviations from the optimum over 25 trials as a function of the CPU time, over 10000 seconds. We consider u1060, for which the CGS algorithms have obtained good results in the previous experiments, and u2319, for which poor results have been obtained. For u1060, the CGS algorithms outperform the ACO algorithms during the entire interval. Although CGS-TSP-C performs best in the initial phases, it is outperformed by CGS-TSP in the long run. In the case of u2319, the CGS algorithms are not able to reach the performance of ACO.

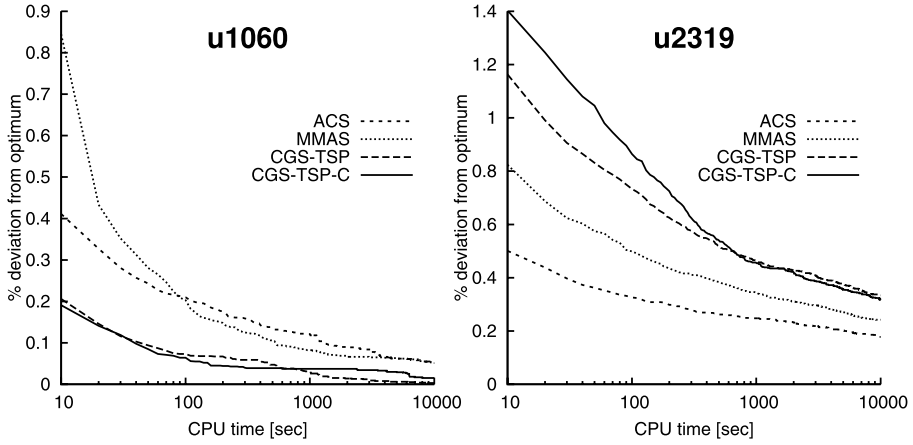

Fig. 2. The development of the mean percentage deviations over 25 trials

4 Conclusions and Future Work

Experimental results indicate that CGS is still able to compete with ACO when the algorithms are combined with local search, although the performance improvement in the case of CGS is not as significant as in the case of ACO. The CGS-TSP-C algorithm proposed in this paper shows that correlating the recommendations with a level of confidence may improve the results. Still, more research is needed in order to determine in which cases CGS-TSP-C should be preferred to CGS-TSP.

Although for our experimental setup the CGS algorithms generally outperform the ACO algorithms, there are a few cases where the performance of CGS is relatively poor. The most striking example in our experiments is the TSP instance u2319. It is therefore worthwhile to investigate which characteristics of the TSP instances influence the performance of the CGS algorithms and how these characteristics relate to the values of the different parameters used by these algorithms. In the case of ACO, it has been shown [6] that an increase in the standard deviation of the cost matrix of TSP instances leads to a decrease in the performance of the algorithm. Like in ACO, the decisions taken in the solution construction phase of CGS algorithms depend on the relative lengths of edges in the TSP. Therefore, we expect that the standard deviation of the cost matrix also affects the performance of these algorithms.

A drawback of CGS algorithms is the large number of parameters to be tuned. We plan to devise a variant of CGS-TSP with only a small number of parameters. One way to achieve this is to identify parameters whose optimal value is not problem specific and to remove these parameters by hard-coding their values in the algorithm.

References

1. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Proceedings of GECCO 2002, pp. 11–18 (2002)
2. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
3. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
4. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An Automatic Algorithm Configuration Framework. Journal of Artificial Intelligence Research (JAIR) 36, 267–306 (2009)
5. Iordache, S.: Consultant-Guided Search - A New Metaheuristic for Combinatorial Optimization Problems. In: Proceedings of the 2010 Genetic and Evolutionary Computation Conference (GECCO 2010). ACM Press, New York (2010)
6. Ridge, E., Kudenko, D.: Determining whether a problem characteristic affects heuristic performance. A rigorous Design of Experiments approach. In: Recent Advances in Evolutionary Computation for Combinatorial Optimization, vol. 153, pp. 21–35. Springer, Heidelberg (2008)
7. Stützle, T., Hoos, H.H.: MAX-MIN Ant System. Future Generation Computer Systems 16(8), 889–914 (2000)