

Does PRNG Impact on Stochastic Local Search? *

Qiang Lv

School of Computer Science and Technology,
Suzhou University
Suzhou, Jiangsu, 215006, P.R. China.
qiang@suda.edu.cn

Weilei Wang

School of Computer Science and Technology,
Suzhou University
Suzhou, Jiangsu, 215006, P.R. China.
210513012@suda.edu.cn

Xiaoyan Xia

Provincial Key Lab of Computer Information Processing Technology
Suzhou, Jiangsu 215006. P.R. China
xyxia@suda.edu.cn

Abstract

When tackling many NP-hard combinatorial optimization problems, a lot of excellent metaheuristics must integrate some kind of strategy of stochastic local search (SLS). In literature, different PRNGs (pseudo-random number generator) are considered to have the same impact on SLS. By comparing and analyzing the computing results of different PRNGs being applied to two successful SLS algorithms, this article gives the evidence to show that different PRNGs have different impact on SLS. By saying “different impact” we mean that such difference can not be simply explained as a phenomenon of chance. In terms of results and performance such difference can be obviously observed as a fact of inevitability in statistical view. We achieve this conclusion by two empirical studies. The first one is to compare and analyze the different behaviors when a SLS, 3Opt, being applied to the Traveling Salesman Problem (TSP) with different PRNGs. The second is about different behaviors when another SLS, Reactive Local Search, being applied to the Maximum Clique Problem. The secondary contribution of this paper is, if the above impact being acknowledged, to find a good PRNG for 3Opt-TSP. And our work illustrate that such PRNG seems to exist.

1 Introduction

For many combinatorial optimization problems, it is comparatively easier to find feasible solutions than to find the optimum, especially for those NP-hard problems.

Metaheuristics are usually considered practically successful approaches to tackle these problems. Such examples of most popular and successful metaheuristics are Genetic Algorithm [19, 17], Stimulated Annealing[23, 10], Tabu Search[15, 16] and Ant Colony Optimization[12, 13]. Local-search-based metaheuristics[7] usually try to do the optimization search in a reduced search space, which is called neighborhood. The neighborhood is what the *local* means in the term local-search. It is natural that even local search cannot ensure to enumerate all the solutions in the neighborhood in reasonable time. Therefore a lot of approximation techniques are always employed to search the neighborhood, e.g, 2Opt approximates to 2-exchange and 3Opt to 3-exchange[20]. Such approximations generally take the usage of a pseudo-random number generator (PRNG), if no other prior knowledge can be taken into guiding the local search. So the strategy of local search always results in some stochastic property of the performance when running the search algorithms. This is what stochastic local search (SLS) is meaning in this paper. We shall not confuse the same term in literature[20] although the two terms are almost in general common, but the latter is with a more general coverage.

In literature, different PRNGs are considered to have the same impact on SLS[20]. By comparing and analyzing the computing results of different PRNGs being applied to SLS, this article gives the evidence to show that different PRNGs have different impact on SLS. By saying “different impact” we mean that such difference can not be simply explained as a phenomenon of chance. In terms of results and performance such difference can be obviously observed as a fact of inevitability in statistical view. We happened to reveal this phenomenon when we were trying to optimize ACOTSP package[1]. This package is about ap-

*Supported by the National Science Foundation of Jiangsu Province of China under Grant No. BK2003030.

plying ACO to the Traveling Salesman Problem (TSP)[11] sequentially combining a local search named 3Opt, which has been proved as a good approximation of the best local search for TSP[18]. In this paper, we call this combination solution 3Opt-TSP.

3Opt-TSP firstly generates a random permutation, whose length is the number of the cities of the TSP instance, by a PRNG. Each position in the permutation contains a city number for the subsequent local optimization. Then 3Opt-TSP tries to optimize the current solution by checking the associated edges based on positions one by one in the permutation. It is obvious that different permutation will result in different final local optimization. This paper uses different PRNGs to generate such permutations. The empirical computing results show that different PRNGs have different impact on 3Opt-TSP. And consequently we try to find a good PRNG for 3Opt-TSP. Finally we provide another computing results when another SLS, reactive local search (RLS), solving the Maximum Clique Problem (MCP) to show that the different impact also does exist for RLS-MCP solution.

This paper is organized as follows: Section 2 gives a very brief introduction of related work because there are little discussions on our topic in literature. Section 3 describes the empirical approach of this paper and how we evaluate the different impact. Section 4 is the main body of this paper, which gives an empirical study on how different PRNGs impact on 3Opt-TSP. Section 5 provides another empirical evidence which shows such different impact exist. We conclude our work in the final Section 6.

2 Related Work

The research of metaheuristics now aims at not only empirically solving specific problems but also theoretically predicating the possibility and performance of the running before a metaheuristic algorithm is applied to solve a problem. The theory of fitness landscape[32, 33, 30] can be used to describe the feature of the target problems. So the analysis of fitness landscape can be helpful to design good metaheuristic algorithms for certain types of problems[31, 8, 9]. Therefore the impact of landscape on search algorithms has been acknowledged and well studied in literature[21, 22, 14]. But since traditional perceive takes it for granted that different PRNGs have the same impact on SLS[20], literature seldom mentioned and studied the impact of PRNGs on SLS.

3 Evaluation Criteria and Running Platform

The empirical study approach of this paper is like this: for the same running platform, keep the same running parameters for ACOTSP, replace the PRNG of ACOTSP with

different PRNGs and test, analyze the ACOTSP running on the problem instances from TSPLib[2].

Now the brief introduction of the running parameters is summarized as follows:

- num-ants: number of ants, the default value is 25.
- num-neighbour: number of neighbour cities, the default value is 20.
- nn-list: number of neighbour cities of each possible exchanged city in 3Opt-TSP, the default value is 20.
- max-tries: the maximum try number for every run, the default value is 20. Due to the stochastic feature, the computing results should be evaluated in some kind of statistic way. So each run usually consists of several repeats. Each such repeat is called a try.
- Alpha, Beta: the impact factor of pheromone and heuristic information respectively. The default setting is 1.0 and 2.2.
- rho: evaporation factor of pheromone, the default value is 0.5.

For more detail semantic information about these parameters, please refer to the ACOTSP package.

The running platform is: IBM P550 with 4 SMP 1.5GHz powerPC CPU, 6GB memory and SuseLinux. The compiler is gcc version 3.3.3.

In order to evaluate how different PRNGs impact on 3Opt-TSP, we must collect statistic data. As what ACOTSP package did, Table 1 summarizes what we collected in the running.

Table 1. Statistic items collected for evaluating different PRNGs impact on 3Opt-TSP

item	explanation
optimum	the optimum value given by the benchmark
b_{avg}	the average of best solutions of all tries
best	the best solution among all the tries
e_b	the excess from the best solution to the optimum
b_{std}	standard deviation of all the best solutions
t_{avg}	average time for getting all the best solutions
t_{std}	standard deviation of time of all tries
$Iter_{avg}$	average iterations for getting all the best solutions
$Iter_{std}$	standard deviation of iterations of all the tries
worst	the worst solution of all the bests among all the tries
$total_{avg}$	the average time of all the tries terminated
$total_{std}$	standard deviation of time used by all tries
e_w	the excess from the worst to the optimum
e_{avg}	the excess from b_{avg} to the optimum

Based on the collected data in Table 1, the evaluating criteria of different PRNGs' behaviors in this paper, suggested by [28], are summarized as follows:

1. If in a given same running time, only method A finds the optimum in all tries on a specific problem instance, then method A is dominant over other methods on that instance.
2. If in a given same running time all the runs of different methods can find the optimum, but only method A can get it in all tries, then method A is the best among the competitors.
3. If in a given same running time, all the runs of different methods can find the optimum but not in every try, then e_{avg} , e_b and e_w are used to rank the winner. If method A wins in all the three performances, then method A is better than the others. If no method wins in all the above three performances in one run, repeat several runs of each method. The method which wins in average performances of the three is judged the winner.
4. If no method can be judged as the winner based on the above criteria, then all the methods are equivalently well.
5. If method A is excellent on every tested problem instance, then it is generalized to behave well on other untested problem instances.

We think the above criteria can be adopted for evaluating other metaheuristics which show stochastic features probably due to taking a PRNG as an algorithm component.

Regarding to the properties of PRNG which might be factors of impacting on 3Opt-TSP, we just consider the following properties:

- Period: the length of a complete sequence of random numbers generated by a PRNG without periodic repetition.
- Speed: the speed of a PRNG producing a random number.
- Distribution: This might be the hardest property to analyze since it involves the definition of what is a random number? In this paper we tend to think all the tested PRNGs here have the same uniform distribution. But at the conclusion part of this paper, we suggest that we should derive other property, similar to distribution here, for a PRNG in terms of measuring how a PRNG matches the fitness landscape of the neighborhood.

4 Analysis of Impact of PRNGs on 3Opt-TSP

4.1 Does Period of PRNG Impact on 3Opt-TSP?

The most simple and typical PRNG is based on linear congruential method[24]. That is the one which follows the equation (1)

$$X_{i+1} = (aX_i + c) \bmod m \quad i = 0, 1, \dots \quad (1)$$

The main parameters are a , c and m . Only when a , c and m are properly chosen, the period of the PRNG will be m . We name the PRNG, which has the parameter settings like $a = 137$, $m = 256$, $c = 187$ for equation (1), method T1 (with the period 256); and the one which ACOTSP package adopts method A (with period 2147483647). Method A will be described in next section. So we have a large gap of the period between these two methods. Apart from the period, the difference between method A and T1 is their generating speed. Our test shows that the ratio is A:T1=1:1.9, that is method T1 is much faster than method A. For the TSP problem instance lin318, the two different PRNGs result in different behaviors of ACOTSP, summarized in Table 2 (the running time is 100 seconds; the bold figures indicate the best result.):

Table 2. Method A/T1 behave differently on lin318

method	b_{avg}	$Iter_{avg}$	b_{std}	$Iter_{std}$	best	worst
A	42034.2	1548.6	16.01	1180.47	42029	42082
T1	42029	3357.95	0	407.32	42029	42029
t_{avg}	t_{std}	$total_{avg}$	$total_{std}$	e_b	e_{avg}	e_w
34.19	26.11	36.98	30.64	0	0.000124	0.001261
80.58	9.6	80.58	9.6	0	0	0

From Table 2 we can see that method T1 always finds the optimum in every try, while method A does not. Please note that neither method T1 or A consumes all the given running time. But method A guides 3Opt-TSP nothing to do further search even if it can not find the optimum, while method T1 tells 3Opt-TSP to do more iterations and as a result it finds the optimum in very try. So we say that method T1 impacts more positively on 3Opt-TSP on instance lin318 than method A.

More problem instances are tested to compare method T1 and A. Table 3 lists the results.

We just list the three items e_b , e_{avg} and e_w for convenience. From Table 3 we can see that method T1 still behaves well on small scale TSP instances, but method A wins on thousand-city problem instances. It seems that the time saved by method T1 does not help a lot for 3Opt-TSP to do more efficient search. It is the period of method T1 who restricts 3Opt-TSP getting more chance to optimize the input solution.

Table 3. Method A/T1 behave on other TSP instances

Instance	method	e_b	e_{avg}	e_w
a280	A	0	0.000124	0.001261
	T1	0	0	0
pr299	A	0	0.000109	0.000809
	T1	0	0	0
gr666	A	0.00001	0.000481	0.001383
	T1	0	0.000438	0.000938
fl1557	A	0.005134	0.007082	0.009458
	T1	0.00572	0.008032	0.009818
d2103	A	0.006704	0.007268	0.008055
	T1	0.006491	0.007438	0.009693
pcb3038	A	0.009064	0.015772	0.023712
	T1	0.013421	0.018327	0.022841

We conclude that, from this set of experiments, the period of PRNG will impact differently on 3Opt-TSP, either positively or negatively. In section 5 we will give additional evidence to support the conclusion given here.

4.2 How Does the Period of PRNG Impact on 3Opt-TSP?

Another common PRNG is produced by multiplicative congruential method[26], which follows

$$I_{i+1} = aI_i \bmod m \quad i = 0, 1, \dots \quad (2)$$

to produce random numbers. We name another PRNG, which has the parameter settings like $a = 23, m = 10^8 + 1$ for equation (2), method T2. Obviously the period of method T2 is a little bit smaller than that of method A. Again method A and T2 applied to 3Opt-TSP behave differently in Table 4(the running time is 100 seconds):

Generally speaking, method A and method T2 behave equivalently well in statistic way. A slight difference of the period is not enough to cause the different behaviors when 3Opt-TSP is running. If we expect a “good” PRNG for 3Opt-TSP, only the period property of a PRNG does not seem to be the deterministic factor.

4.3 Period and Speed of PRNG Impact on 3Opt-TSP

It seems that larger period with faster speed of PRNG will do well to 3Opt-TSP. Let’s make such PRNG by combining two PRNGs to enlarge the period[25]. Let sequence $q_1 = x_1^a, x_2^a, \dots, x_{m_1}^a, \dots$, with the period m_1 , and sequence $q_2 = x_1^b, x_2^b, \dots, x_{m_2}^b, \dots$, with the period m_2 , for any $c_a, c_b > 0$, by the following equation

$$U_i = \frac{c_a x_i^a}{m_1} + \frac{c_b x_i^b}{m_2}, u_i = U_i \bmod 1, \quad i = 1, 2, \dots \quad (3)$$

Table 4. Method A/T2 behave on some TSP instances

Instance	method	e_b	e_{avg}	e_w
a280	A	0	0.000039	0.000775
	T2	0	0.000194	0.000775
2nd run	A	0	0.000213	0.00349
	T2	0	0.000078	0.000775
lin318	A	0	0.00036	0.001475
	T2	0	0.000163	0.001475
gr666	A	0.000034	0.000624	0.001274
	T2	0	0.000505	0.000968
fl1557	A	0.004008	0.007411	0.009593
	T2	0.004864	0.007026	0.010313
pr2392	A	0.003238	0.005115	0.009743
	T2	0.002672	0.004331	0.009346

we can produce a new sequence u_i with a larger period. For example, when $m_1 = 2147283563$ and $m_2 = 2147483399$, the new sequence will have the period 2.3×10^{18} . We name such PRNG method D.

Since method D is slower than method A, Knuth proposed a speed up technique[24]. Based on the following formula:

$$X_n = (X_{n-55} - X_{n-24}), 55 \leq n \leq 79 \quad (4)$$

given by an initial value as seed, construct a table with 56 “random” numbers. By looking up the table, we can easily produce random number with a very fast speed. We refer the readers to [27] for the implementation. We name this PRNG method E. The speed ratio of the three methods is A:D:E=1:0.6:2[27].

The results of these different PRNGs impacting on 3Opt-TSP on small TSP instances are listed in Table 5(running time is 100 seconds):

From Table 5 we can not draw any conclusions about which method behaves best. Method D has much larger period but with half speed of method A. Method E also has much larger period but with double speed of method A. But both method D and E do not behave well or badly comparing with method A. It seems that period and speed are still not enough for a PRNG to impact on 3Opt-TSP.

How about distribution property? The discussion of distribution is beyond the scope of this paper. We have to pretend to think that all the PRNGs have the same distribution.

4.4 A Good PRNG to 3Opt-TSP

Does a good PRNG exist for 3Opt-TSP? Let’s now check the method A mentioned in the above subsections, which 3Opt-TSP adopts.

In real world, a most popular PRNG is based on equation (2) which has the following settings[26]: $a = 7^5 = 16807$,

Table 5. Method A/D/E behave on small scale TSP instances

Instance	method	e_b	e_{avg}	e_w
lin318	A	0	0.00009	0.001285
	D	0	0.000482	0.002379
	E	0	0.00025	0.002237
2nd run	A	0	0.000319	0.001475
	D	0	0.000359	0.001475
	E	0	0.000193	0.002379
ali535	A	0	0.000206	0.000534
	D	0	0.000188	0.000534
	E	0	0.000143	0.000366
p654	A	0	0.000289	0.000577
	D	0	0.000303	0.000635
	E	0	0.000277	0.000577
2nd run	A	0	0.000281	0.000491
	D	0	0.000315	0.000693
	E	0.000173	0.000374	0.000577

$m = 2^{31} - 1 = 2147483647$. But the implementation of this PRNG is not efficient for 32-bit machine because $a \times (m - 1)$ usually is much much larger than a 32-bit integer. So a practical PRNG is generated by decomposing m [29]: Let $m = aq + r$, i.e., $q = \lfloor m/a \rfloor$, $r = m \bmod a$, with square brackets denoting integer part. If r is small, specifically $r < q$, and $0 < z < m - 1$, it can be shown that both $0 \leq a(z \bmod q) \leq m - 1$ and $0 \leq r \lfloor z/q \rfloor \leq m - 1$, and that

$$az \bmod m = \begin{cases} a(z \bmod q - r \lfloor a/q \rfloor), & \text{if it is } \geq 0; \\ a(z \bmod q - r \lfloor a/q \rfloor) + m, & \text{otherwise.} \end{cases} \quad (5)$$

So by choosing proper parameters a, q and r formula (2) will produce a nice random number sequence. We name such a PRNG, with such settings $a = 16807, q = 127773$ and $r = 2836$, method A. This is the PRNG which ACOTSP package adopts. Similarly we name other two PRNGs with different settings: method B with $a = 48271, q = 44488, r = 3399$ and method C with $a = 69621, q = 30845, r = 23902$.

Let's begin with some simple TSP instances in Table 6(the running time is 100 seconds):

Table 6 repeats tests instance a280 for 3 runs, att532 for 2 runs, lin318 for 2 runs. Table 6 says that method B dominates over other methods on the three instances.

How about the larger instance? Let's study the individual TSP instance d2103 in more detail. Since none of method A, B or C can find the optimum, let's repeat 5 runs for testing all the three methods on instance d2103. Table 7 lists the results:

Table 7 just states that method B is better than the other two variants on this problem instance. Table 8 averages the

Table 6. Method A/B/C behave differently on small scale TSP instances

Instance	method	e_b	e_{avg}	e_w
a280 (10s)	A	0	0.000116	0.000775
	B	0	0	0
	C	0	0.000562	0.008143
2nd run	A	0	0.000078	0.000775
	B	0	0	0
	C	0	0.000155	0.001551
3rd run	A	0	0.000523	0.008143
	B	0	0.00033	0.00349
	C	0	0.001047	0.010081
Att532 (100s)	A	0	0.000706	0.001806
	B	0	0.000477	0.001011
	C	0.00065	0.000858	0.00112
2nd run	A	0.000253	0.000585	0.001517
	B	0	0.000511	0.001625
	C	0	0.000518	0.0013
lin318 (100s)	A	0.000253	0.000591	0.001336
	B	0	0.000455	0.001084
	C	0	0.000435	0.000975
2nd run	A	0	0.000214	0.002165
	B	0	0.000074	0.001475
	C	0	0.00029	0.001475

Table 7. Method A/B/C behave on instance d2103

Instance	method	e_b	e_{avg}	e_w
d2103 (100s)	A	0.006917	0.007745	0.00893
	B	0.006616	0.007399	0.009018
	C	0.006554	0.007579	0.010194
2nd run (100s)	A	0.006491	0.007399	0.009143
	B	0.006229	0.007426	0.00878
	C	0.006554	0.007579	0.010194
3rd run (100s)	A	0.006291	0.007416	0.008668
	B	0.006654	0.007399	0.009518
	C	0.006566	0.00751	0.008955
4th run (200s)	A	0.006429	0.007042	0.007592
	B	0.006366	0.006847	0.007805
	C	0.006729	0.007171	0.007905
5th run (300s)	A	0.006291	0.006869	0.007342
	B	0.006229	0.006821	0.007755
	C	0.006729	0.00731	0.007905

results of Table 7.

Table 8. Average of method A/B/C tested on d2103

method	e_b	e_{avg}	e_w
A	0.00648	0.00729	0.00834
B	0.00642	0.00718	0.00858
C	0.00663	0.00743	0.00903

Table 8 shows that in statistic view, method B is better than method A and C.

Table 9 summarized the different results of method A, B and C tested on more TSP instances. Basically Table 9 shows the same conclusion which Table 7 and 8 reveal.

Now it seems to us that we would like to draw the conclusion that method B is the best PRNG for 3Opt-TSP upon the tested problem instances.

5 Impact of PRNGs on RLS-MCP

RLS is based on local search complemented by a feedback scheme to enhance the diversification. MCP is an NP-hard problem with strong negative results on its approximation properties[4]. Battiti proposed a RLS algorithm to solve MCP[6], which has been considered one of the best state-of-art solutions to MCP. Although the recent studies have improved this state-of-art to a higher level[28], Battiti quickly responded it with more categories of problem instances[5]. The empirical study shows that RLS still remains in the state-of-art in terms of robustness and speediness. The test instances of the next part are all the same as the ones tested in [6], from DIMACS challenges[3].

We compare two different PRNGs here, one is method T2, the other is from RLS-MCP package, which is a regular gcc library implementation. We name the latter PRNG as method MO. All the runs are terminated by the same criteria: getting the best record, time out or reaching the specific maximum iterations. Typical 36 DIMACS MCP instances were tested in [6]. We also run the RLS-MCP, with the two different PRNGs, on the same 36 instances. 27 of them have the same behavior for this RLS-MCP with different PRNGs. But for other 9 instances, interesting results are listed in Table 10.

In Table 10, Instance column shows the problem instance name, with running seconds in parentheses; br column stands for the best record of the maximum clique number, with a star for saying this is a proved optimum. The running platform is the same as the one on which 3Opt-TSP is tested.

It is shown in Table 10 that method T2 is prior to method MO on 7 of the 9 instances in terms of both solution quality

Table 9. Method A/B/C tested on more TSP instances

Instance	method	e_b	e_{avg}	e_w
gr666 (100s)	A	0	0.000511	0.000992
	B	0	0.000421	0.001271
	C	0	0.000412	0.001155
2nd run	A	0	0.000485	0.000989
	B	0	0.00044	0.000917
	C	0	0.000474	0.001029
3rd run	A	0	0.000538	0.001355
	B	0	0.000571	0.001257
	C	0.000605	0.000839	0.001383
4th run	A	0	0.000477	0.00178
	B	0	0.000371	0.000883
	C	0	0.000569	0.002099
pr299 (50s)	A	0	0.000108	0.000685
	B	0	0.000098	0.001017
	C	0	0.000108	0.000685
2nd run	A	0	0.0002	0.001826
	B	0	0.000189	0.001598
	C	0	0.000285	0.001826
3rd run	A	0	0.000146	0.001473
	B	0	0.000204	0.001349
	C	0	0.000227	0.001038
4th run	A	0	0.000194	0.001764
	B	0	0.000081	0.000706
	C	0	0.000173	0.000913
fl3795 (100s)	A	0.003236	0.005254	0.008315
	B	0.003236	0.005016	0.009663
	C	0.003236	0.004695	0.008809
2nd run	A	0.003873	0.00603	0.007881
	B	0.004639	0.005902	0.007926
	C	0.005044	0.006199	0.009638
3rd run	A	0.004594	0.006069	0.008107
	B	0.004233	0.005904	0.007521
	C	0.00572	0.008199	0.012385
p654 (100s)	A	0	0.000268	0.00052
	B	0	0.000255	0.000751
	C	0	0.000328	0.000693
2nd run	A	0	0.000328	0.000693
	B	0	0.00028	0.000635
3rd run	A	0	0.000221	0.000635
	B	0	0.000257	0.000577
4th run	A	0	0.000286	0.00052
	B	0	0.000258	0.00052

Table 10. Two different PRNGs result differently on RLS-MCP

Instance	method	solution	iteration	br
brock200_4 (19.49s)	MO	16	319903	17*
	T2	17	10966	
gen400_p0.9_55 (1.2s)	MO	53	35017	55
	T2	55	1193	
p_hat700-2 (0.028s)	MO	43	140	44*
	T2	44	78	
C250.9 (0.029s)	MO	43	1340	44*
	T2	44	378	
C1000.9 (41.6s)	MO	67	1964	67
	T2	68	379873	
gen200_p0.9_44 (0.037s)	MO	39	1865	44*
	T2	40	1865	
keller5 (0.171s)	MO	26	2990	27
	T2	27	2529	
keller6 (189s)	MO	59	79152	59
	T2	58	1173480	
p_hat700-3 (0.035s)	MO	62	252	62
	T2	61	333	

and running speed. Only 2 of the 9 gives the reverse results. Can this result be interpreted as a chanciness because of stochastic property of the PRNG? We don't think so, although we now can not prove the opposite guess. The purpose of this section is just to provide additional "statistic" evidence to support the presentation of the above sections.

6 Conclusion and Future work

In this paper, we investigate whether different PRNGs have impact on SLS or not. Right now only three properties of a PRNG are considered as the possible factors of such impact. We evaluate all together 8 PRNGs, T1, T2, A, B, C, D, E, MO. And we find that different PRNGs do impact differently on 3Opt-TSP and RLS-MCP on our tested instances. Further more we find that method B is overall better than other methods for 3Opt-TSP.

It is well understood that the fitness landscape of neighborhood deeply impacts on the performance of a SLS algorithm. In our current understanding, the distribution (and other possible unrevealed properties?) of a PRNG will also affect the search behavior of a SLS algorithm. If such influence matches the landscape and search trajectory, the SLS algorithm will benefit a lot when it completes walking the trajectory.

In the near future, the agenda of our work will focus on the following two issues. Firstly we need more empirical studies to provide more evidences to prove that the impact we mentioned does exist. Secondly, we want to derive more properties of a PRNG which might differentiate enough to

analyze the different impact on SLS. We believe that if the impact we proposed exists for general SLS algorithms, it will be an interesting new research topic to choose a "good" PRNG when one designs metaheuristic algorithms. And when evaluating metaheuristics the potential impact from a PRNG should also be considered. We just want to bring up the awareness of detecting the possible impact of PRNG on SLS algorithms.

Researchers usually acknowledge the approach to classifying the problem instances by the fitness landscape in order to design an efficient and robust metaheuristic algorithm. So why not do the similar research work to study how a PRNG matches the landscape?

7 Acknowledgements

Great thanks to Thomas Stützle, the author of ACOTSP package. Same great thanks to Roberto Battiti, the author of RLS-MCP package.

References

- [1] <http://www.aco-metaheuristic.org/downloads/ACOTSP.V1.0.tar.gz>.
- [2] <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.
- [3] <http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/DIMACS/II/descr.html>.
- [4] G. Ausiello, P. Crescenzi, and M. Protasi. Approximation solution of np optimization problems. *Theoretical Computer Science*, 150:1–55, 1995.
- [5] R. Battiti and F. Mascia. Reactive and dynamic local search for max-clique, does the complexity pay off? Technical Report DIT-06-027, Informatica e Telecomunicazioni, University of Trento, 2006.
- [6] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29(4):610–637, 2001.
- [7] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.
- [8] K. Boese. A new adaptive multi-start technique for combinatorial global optimization. *Operations Research Letters*, 16:101–113, 1994.
- [9] K. D. Boese. *Models for Iterative Global Optimization*. PhD thesis, UCLA, 1996.

- [10] V. Cerny. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–45, 1985.
- [11] M. Dorigo, V. Maniezzo, and A. Coloni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):1–13, 1996.
- [12] M. Dorigo and T. Stützle. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, chapter The ant colony optimization metaheuristic: Algorithms, applications and advances., pages 251–285. Kluwer Academic Publishers, Norwell, MA, 1999.
- [13] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Boston, MA., 2004.
- [14] C. Fonlupt, D. Robilliard, P. Preux, and E.-G. Talbi. *Meta-Heuristics – Advances and Trends in Local Search Paradigms for Optimization*, chapter 18 Fitness landscapes and performance of meta-heuristics, pages 255–266. Kluwer Academic Press, 1999.
- [15] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Science*, 8:156–166, 1977.
- [16] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publisher, 1997.
- [17] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine learning*. Addison Wesley, Reading, MA, 1989.
- [18] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.
- [19] J. H. Holland. *Adaption in natural and artificial systems*. The University of Michigan Press, Ann Harbor, MI, 1975.
- [20] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann Publishers, 2004.
- [21] T. Jones. *Evolutionary algorithms, fitness landscapes and search*. PhD thesis, University of New Mexico, Albuquerque, NM, 1995.
- [22] T. Jones. One operator, one landscape. Technical Report 95-02-025, Santa Fe Institute, 1995.
- [23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
- [24] D. Knuth. *The art of computer programming*, volume 2, chapter 3 Random Numbers. Addison-Wesley publishing company, 1981.
- [25] P. L’Ecuyer. Efficient and portable combined random number generators. *Communications of the ACM*, 31(6):742–774, 1988.
- [26] S. Park and K. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31(5):1192–1201, 1988.
- [27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*, chapter 7 Random Numbers. Cambridge University Press, 1988.
- [28] W. Pullan and H. H. Hoos. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research*, 25:159–185, 2006.
- [29] L. Schrage. A more portable fortran random number generator. *ACM Transactions on Mathematical Software*, 5(2):132–138, 1979.
- [30] P. F. Stadler. Towards a theory of landscapes. In R. López-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche, editors, *Complex Systems and Binary Networks*, volume 461, pages 77–163, Berlin, New York, 1995. Springer Verlag.
- [31] T. Stützle and H. Hoos. Max-min ant system. *Future Generation Computer Systems Journal*, 16(8):889–914, 2000.
- [32] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D. F. Jones, editor, *International Proceedings of the Sixth International Congress on Genetics*, volume 1, pages 356–366, 1932.
- [33] S. Wright. Surfaces of selective value. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 58, pages 165–172, 1967.