



International Journal of Engineering, Business and Enterprise Applications (IJEBA)

www.iasir.net

Performance Evaluation and Comparison of GA, SA & LSA Based Algorithms for Standard Cell Placement in VLSI Design

Dr. Aaquil Bunglowala¹, Dr. Nidhi Asthana²

Department of Electronics and Telecommunication¹, Department of Engineering Mathematics²

MPSTME, Shirpur Campus, Maharashtra¹, SAIT, Indore, M.P.²

INDIA

Abstract: This paper deals with the concept of hybridization and reports application of hybridizing on GA, SA and LSA. These algorithms were independently used to solve the standard cell placement problem [3,4,5]. In the first section we present a new hybrid technique of SA and GA named Parallelized SA & GA (SAGAP) of different versions for SCP problem. Then we discuss a hybrid technique of GA and LSA named Memetic Algorithm (MA). We compare the results of wire length and CPU time in association with SCP problems from hybrid systems of SA, GA and LSA

Keywords: NP Hard, Standard Cell Problem, Simulated Annealing, Genetic Algorithm, Local Search Algorithm, Memetic Algorithm

I. Introduction

Placement is the most crucial problem during physical design stage. It is accountable for minimizing the area of the chip and interconnection wire-length. Therefore, placement is the key step in minimizing the fabrication cost per chip and maximizing its performance.

The standard cell problem is stated as: Given an electrical circuit consisting of fixed rectangular shaped cells and a net-list stating interconnections among terminals on the periphery of the cells and on the periphery of the circuit itself, it is required to construct a layout indicating the position of each cell such that all the nets can be routed and the total area is minimized. The objective for high performance systems is to minimize the total delay of the system by minimizing the length of the critical paths [10]. The quality of placement is based on layout area, completion of routing and circuit performance. SCP is computationally NP-hard. These problems cannot be solved in polynomial time.

II. Hybrid techniques of SCP

A. Hybrid of SA and GA for SCP

SA is a stochastic heuristic optimization method. It starts with a high temperature and an initial feasible solution. A GA performs an adaptive search analogous to the natural evolution process. A simple GA starts with a set of fixed number of population. SA does not use powerful operators like a cross-over in GA. The combined GA and SA methods can be categorized into two types.

1. Just couple a SA and a GA.
2. Some features from both methods and integrate them.

A new hybrid of SA and GA named Parallelized SA & GA (SAGAP) is introduced. Any hybrid method combines good features of SA and GA and avoids their drawbacks. The method was originally proposed by Mahfoud and Goldberg and named parallel re-combinative SA [7,8]. SAGAP inherits the global convergence property of SA and explicit and implicit parallelism of GA. It takes the population concept from GA and the temperature concept from SA. SAGAP starts with a set of individuals (population) chosen randomly and with a high temperature. It gives the strong genetic operators, cross-over and mutation, to create additional children.

B. Parallelization of SAGAP

The first implementation of SAGAP, on a parallel machine, has been carried out by Mahfoud and Goldberg [7,8]. In the parallel version of SAGAP, one algorithm runs on each node and periodically the solutions are exchanged between them. This process is called migration. On each node, the temperature is set at a sufficiently high value and the initial population is constructed. Then two individuals are selected for CXO followed by the mutation. When the new population is constructed, some individuals are randomly migrated to other nodes. the

SAGAP algorithms inherently parallel and its implementation on a parallel machine is straight forward. An algorithm outlined in figure 2.1 is incorporated for the proposed SAGAP. Cycle cross-over (CXO) operator is taken for GA.

Figure 1: SAGAP algorithm for cell placement

```

Set temperature Temp to a sufficiently high value
Create a population of size  $N_{pop}$  by generating  $N_{pop}$  consistent placements
    Calculate x and y coordinates for each cell of all individual placements
    Evaluate the fitness value of each individual
REPEAT
    DO  $N_{pop}/2$  times:
        Select two parents and apply CXO creating two new children
        Apply mutation operator to each child once
        Calculate x and y coordinates for each cell of the children
        Evaluate the fitness value of the children with regard to the objective function
        Select two parents and the two children by Boltzmann trial
        Build new  $N_{pop}$  by replacing old individuals by newly selected ones
    Lower the temperature Temp by fixed cooling rate
UNTIL termination criteria
  
```

C. Hybrid of GA and LSA for SCP

Memetic algorithms (MAs) are extension of GA with the introduction of individual learning as an additional process of local refinement to accelerate local search. Recent studies of MA have demonstrated that they converge to high-quality solutions more efficiently than the conventional counterparts [1, 2, 6, 9, 11, 12] on many real world applications. Memetic algorithms implements split local search technique to optimize the fitness of individuals by hill climbing. MA combines global and local search by using GA to perform exploration whereas the local search techniques are applied for exploitation [1]. One of the main objectives in implementing any MA is in achieving the best of both techniques during the search. A large number of the local searches techniques immerse within the MAs are not standard; they intend to perform a shorter truncated local search.

Figure 2: Memetic algorithm for cell placement

```

Set population size, generation size
Set mutation rate and cross-over rate
Set generation=0

Generate initial population  $N_{pop}$  randomly
REPEAT
    Select individuals for mating
    Apply cross over
    Apply mutation
    Apply Hill-Climbing to  $N_{pop}$ 
UNTIL (generation  $\leq$  generation size)
  
```

III. Comparison and results

A. Test results of hybrid of SA and GA (SAGAP) for SCP

In this subsection three population sizes of 16, 32 and 64 are used with all the selection policies (SP_I , SP_{II} and SP_{III}). The versions of selection based SAGAP used are:

Table 1

SAGAP	Description	Population size
S-I	SP_I (each child against one of its own parent) with population size	16
S-II	SP_I with population size	32
S-III	SP_I with population size	64
S-IV	SP_{II} (each child against one parent excluding its own parents)	16
S-V	SP_{II} with population size	32
SAGAP	Description	Population size
S-VI	SP_{II} with population size	64
S-VII	SP_{III} (good child against bad parent) with population size	16
S-VIII	SP_{III} with population size	32
S-IX	SP_{III} with population size	64

In table 2 Wire Lengths for different versions are solved by selection based SAGAP. As shown in table 3 CPU time for different versions is solved by selection based SAGAP. We observe in table 2 and 3, the version SAGAP S-VI offers best results in all the test cases except for the test case A where SAGAP S-III offered better average value of wire length.

With these results we deduce that the selection policy whereby each child contends against one parent excluding its own parent randomly with a population size of 64, presents best results as compared to other combinations. Note that with the increase in population size, the algorithm becomes sluggish. Same selection policy with smaller population is quite fast, but the wire length increases by about 1.5%. So, we conclude that for best results with no time constraint SAGAP S-VI must be preferred.

Table 2: Wire Lengths for different versions of selection based SAGAP

Test Case	Result Quality	Wire Length (μm) SAGAP								
		S-I	S-II	S-III	S-IV	S-V	S-VI	S-VII	S-VIII	S-IX
A	Best	3711	3705	3707	3715	3725	3688	3718	3712	3715
	Worst	3977	3842	3752	3844	3814	3789	4147	4012	3811
	Average	3826	3798	3720	3774	3750	3754	3855	3827	3774
B	Best	2822	2793	2763	2805	2775	2751	2894	2865	2763
	Worst	2907	2852	2814	2871	2831	2797	3023	2968	2812
	Average	2864	2819	2786	2832	2797	2775	2958	2913	2784
C	Best	2744	2736	2728	2732	2728	2727	2752	2744	2732
	Worst	2898	2893	2762	2844	2796	2727	2904	2898	2845
	Average	2832	2816	2748	2792	2768	2727	2849	2833	2792
D	Best	3548	3537	3467	3516	3496	3437	3560	3549	3517
	Worst	3703	3644	3526	3609	3575	3475	3762	3704	3610
	Average	3646	3601	3512	3575	3549	3473	3692	3647	3575
E	Best	2545	2554	2536	2545	2536	2535	2536	2545	2545
	Worst	2722	2688	2583	2659	2630	2535	2757	2723	2659
	Average	2658	2628	2565	2611	2594	2535	2689	2659	2611
F	Best	2523	2573	2473	2522	2473	2472	2474	2523	2523
	Worst	2652	2607	2513	2579	2552	2472	2698	2653	2580
	Average	2615	2583	2494	2548	2514	2472	2647	2615	2549

Table 3: CPU time for different versions of selection based SAGAP

Test Case	Result Quality	CPU Time (seconds) SAGAP								
		S-I	S-II	S-III	S-IV	S-V	S-VI	S-VII	S-VIII	S-IX
A	Best	103	154	349	888	1805	4332	120	259	748
	Worst	137	157	360	939	2410	4492	137	292	998
	Average	113	156	355	908	2105	4432	126	267	872
B	Best3	110	149	344	897	1682	4242	133	267	697
	Worst	141	154	352	947	2124	4322	144	304	880
	Average	117	152	348	914	1888	4292	138	281	782
C	Best	111	144	342	901	1657	4192	137	273	687
	Worst	153	149	351	960	2085	4252	150	305	864
	Average	126	148	345	935	1859	4222	142	283	770
D	Best	239	484	1066	2364	5257	13417	227	868	2178
	Worst	254	498	1124	2541	7800	14199	263	942	3231
	Average	247	490	1104	2421	6064	13788	248	887	2512
E	Best	229	476	1039	2227	5203	13297	220	854	2156
	Worst	252	500	1105	2434	6260	13858	256	895	2594
	Average	243	491	1066	2362	5874	13587	237	868	2434
F	Best	222	461	1013	2169	5174	13176	216	839	2143
	Worst	239	471	1024	2468	5759	13718	245	890	2386
	Average	229	463	1019	2361	5380	13467	229	862	2229

B. Test results of hybrid of GA and LSA (MA) for SCP

In this subsection, we present results of MA for different parameter settings like population size, cross over rate, mutation rate and number of generations as shown in table 4. It is noteworthy that application of local search before and after GA produce improved results. The results improve with population size. The results presented in the table show that the wire lengths are reduced as much as 12-15% in case of MA-IX. Note that the population size here is 64 in contrast to 132 in case of GA-IV. Also the value of MAXGEN is lesser in case of MAs. Table 5 indicates that the time taken for the MA for completion is almost 20-30% less than that of GA with population size of 64.

Table 4: Results of Memetic Algorithms [wire length in μm]

Test Case	Result Quality	Wire length in μm								
		MA I	MA II	MA III	MA IV	MA V	MA VI	MA VII	MA VIII	MA IX
A	Best	3505	3498	3318	3458	3456	2950	3401	3416	3269
	Worst	3757	3596	3364	3523	3520	2987	3645	3558	3448
	Average	3582	3555	3360	3496	3475	2927	3506	3520	3385
B	Best	2712	2644	2511	2632	2640	2281	2586	2543	2440
	Worst	2844	2735	2598	2738	2746	2353	2664	2584	2480
	Average	2769	2680	2545	2673	2687	2320	2625	2564	2461
C	Best	2588	2580	2455	2554	2565	2199	2515	2525	2418
	Worst	2694	2688	2554	2646	2624	2245	2656	2598	2418
	Average	2641	2626	2504	2594	2592	2215	2596	2562	2418
D	Best	3368	3357	3188	3327	3309	2849	3253	3224	3047
	Worst	3537	3480	3306	3456	3430	2990	3395	3291	3081
	Average	3477	3433	3257	3405	3381	2935	3343	3270	3079
E	Best	2810	2818	2636	2895	2884	2678	2333	2352	2248
	Worst	2958	2925	2766	3013	2990	2792	2496	2415	2248
	Average	2897	2868	2713	2957	2951	2756	2437	2387	2248
F	Best	2736	2782	2555	2808	2772	2603	2313	2333	2192
	Worst	2876	2832	2690	2924	2911	2715	2432	2348	2192
	Average	2828	2798	2629	2867	2851	2621	2398	2337	2192

Table 5 : Results of Memetic Algorithms [CPU time in seconds]

Test Case	Result Quality	CPU time in seconds								
		MA I	MA II	MA III	MA IV	MA V	MA VI	MA VII	MA VIII	MA IX
A	Best	99	111	209	133	182	249	285	313	352
	Worst	134	154	215	192	276	257	384	431	365
	Average	109	123	212	149	208	254	313	359	360
B	Best	107	122	205	148	208	243	310	353	346
	Worst	119	136	210	167	238	249	342	385	353
	Average	112	127	207	155	219	246	323	365	350
C	Best	102	115	202	140	196	239	300	340	343
	Worst	134	155	208	194	281	244	387	428	350
	Average	118	135	205	166	238	242	341	383	345
D	Best	263	288	642	333	436	774	747	885	1085
	Worst	272	299	672	345	454	814	786	933	1146
	Average	266	292	660	338	443	792	769	907	1118
E	Best	255	278	627	321	418	765	720	864	1067
	Worst	291	322	665	378	505	799	828	970	1122
	Average	277	305	644	356	471	784	780	926	1092
F	Best	234	254	610	289	370	755	665	815	1050
	Worst	250	273	619	315	409	783	706	875	1080
	Average	236	257	614	293	376	769	673	833	1066

IV. Conclusion

This paper investigated in detail the hybrid systems based on SA, GA and LSA Hybrid of LSA and GA has been attempted first and it quite significantly contributed in speeding up the performance of SA. Another hybrid method investigated has been hybridization of GA with SA in SAGAP algorithm. One advantage of this method is its global convergence property inherited from SA approach. MA presents results better than the other hybrid techniques of SAGAP in specific, and moreover proves to be even better than any of the iterative, local search or coupled network approach in general as shown in table 6 and 7 (refer figure 3 & 4) It is true that the parameters set play a major role in defining the performance and results of any hybrid algorithm. Hybrid method investigated has been hybridization of GA with SA in SAGAP algorithm. One advantage of this method is its global convergence property inherited from SA approach. The last hybrid method employing GA and LSA implemented and tested in this work is MA. The performance of MA surprisingly varied with different combinations of the values of mutation and cross-over rates, sometimes giving very poor results and at other occasions gives results far better than any previously established technique. The results, with right parameter sets of this algorithm proved to be the best as compared to any stand alone or hybrid technique.

Table 6: Comparison of average wire length (in μm)

TEST CASES	SAGAP	MA
A	3786.444	3422.89
B	2836.444	2591.56
C	2795.222	2527.56
D	3585.556	3286.67
E	2616.667	2690.44
F	2559.667	2613.44

Figure 3: Comparison of average wire length (in μm)

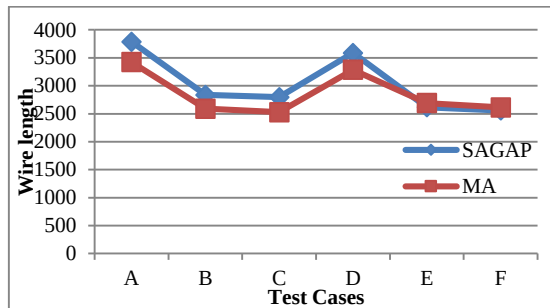
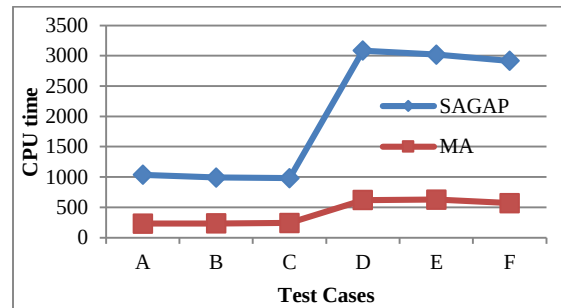


Table 7: Comparison of average CPU time (in seconds)

TEST CASES	SAGAP	MA
A	1037.11	231.89
B	990.222	233.78
C	981.111	241.44
D	3084.56	620.56
E	3018	626.11
F	2915.44	568.56

Figure 4: Comparison of average CPU time (in seconds)



References

- [1] Areibi, S., Bao, X., Grewal, G., Banerji, D., Du, P., "A Comparison of Heuristics for FPGA Placement", ACTA International Journal of Computers and Applications
- [2] A. Vicini and D. Quagliarella, "Airfoil and wing design using hybrid optimization strategies" Amer. Inst. Aeronautics Astronautics J., vol. 37, no. 5, pp. 634–641, 1999.
- [3] Bunglowala, A., Singhi, B. M., "Memetic Algorithms as a Solution to combinatorial Optimization Problem", Proceedings of 2nd PIMR International Conference, 2008.
- [4] Bunglowala, A., Singhi, B.M., "Performance Evaluation And Comparison and Improvement of Standard Cell Placement in VLSI Design", International Conference on Emerging Trends in Engineering and Technology, July 2008
- [5] Bunglowala, A., Singhi, B.M., "A Solution to combinatorial Optimization Problem using Memetic Algorithms", International Journal of Computer System Applications [IJCSA], December 2008.
- [6] C. Houck, J. Joines, and M. Kay, "Utilizing lamarckian evolution and the baldwin effect in hybrid genetic algorithms" Meta-Heuristic Research and Application Group, Depart. Ind. Eng., North Carolina State Univ., Raleigh, CA, NCSU-IE Tech. Report 96-01, 1996.
- [7] Mahfoud, S. W. and Goldberg, D. E.: "Parallel Recombinative Simulated Annealing" IlliGAL Report No. 92002 (Illinois Genetic Algorithms Laboratory), University of Illinois (1992a).
- [8] Mahfoud, S. W. and Goldberg, D. E.: "A Genetic Algorithm for Parallel Simulated Annealing" in Maenner, R. and Manderick, B. (Eds.): "Parallel Problem Solving from Nature" 2 (1992b).
- [9] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms" Caltech Concurrent Computation Program, Publication Report 790, 1989.
- [10] Sherwani, N. A.: "Algorithms for VLSI Physical Design Automation" Boston (1993).
- [11] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling" Amer. Inst. Astronautics J., vol. 41, no. 4, pp. 687–696, 2003.
- [12] Y. S. Ong and A. J. Keane, "Meta-lamarckian learning in memetic algorithms," IEEE Trans.Evol. Comput., vol. 8, no. 2, pp. 99–110, Apr. 2004.