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 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 straightforward. An algorithm outlined in figure 2.1 is incorporated for the proposed SAGAP. Cycle cross-over (C XO) 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<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>
<thead>
<tr>
<th>SAGAP</th>
<th>Description</th>
<th>Population size</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-I</td>
<td>SP_I (each child against one of its own parent) with population size</td>
<td>16</td>
</tr>
<tr>
<td>S-II</td>
<td>SP_I with population size</td>
<td>32</td>
</tr>
<tr>
<td>S-III</td>
<td>SP_I with population size</td>
<td>64</td>
</tr>
<tr>
<td>S-IV</td>
<td>SP_{II} (each child against one parent excluding its own parents)</td>
<td>16</td>
</tr>
<tr>
<td>S-V</td>
<td>SP_{II} with population size</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAGAP</th>
<th>Description</th>
<th>Population size</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-VI</td>
<td>SP_{II} with population size</td>
<td>64</td>
</tr>
<tr>
<td>S-VII</td>
<td>SP_{III} (good child against bad parent) with population size</td>
<td>16</td>
</tr>
<tr>
<td>S-VIII</td>
<td>SP_{III} with population size</td>
<td>32</td>
</tr>
<tr>
<td>S-IX</td>
<td>SP_{III} with population size</td>
<td>64</td>
</tr>
</tbody>
</table>
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

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

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

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Result Quality</th>
<th>S-I</th>
<th>S-II</th>
<th>S-III</th>
<th>S-IV</th>
<th>S-V</th>
<th>S-VI</th>
<th>S-VII</th>
<th>S-VIII</th>
<th>S-IX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Best</td>
<td>103</td>
<td>154</td>
<td>349</td>
<td>888</td>
<td>1805</td>
<td>4332</td>
<td>120</td>
<td>259</td>
<td>748</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>137</td>
<td>157</td>
<td>360</td>
<td>939</td>
<td>2410</td>
<td>4492</td>
<td>137</td>
<td>292</td>
<td>998</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>113</td>
<td>156</td>
<td>355</td>
<td>908</td>
<td>2105</td>
<td>4432</td>
<td>126</td>
<td>267</td>
<td>872</td>
</tr>
<tr>
<td>B</td>
<td>Best3</td>
<td>110</td>
<td>149</td>
<td>344</td>
<td>897</td>
<td>1682</td>
<td>4242</td>
<td>133</td>
<td>267</td>
<td>697</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>141</td>
<td>154</td>
<td>352</td>
<td>947</td>
<td>2124</td>
<td>4322</td>
<td>144</td>
<td>304</td>
<td>880</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>117</td>
<td>152</td>
<td>348</td>
<td>914</td>
<td>1888</td>
<td>4292</td>
<td>138</td>
<td>281</td>
<td>782</td>
</tr>
<tr>
<td>C</td>
<td>Best</td>
<td>111</td>
<td>144</td>
<td>342</td>
<td>901</td>
<td>1657</td>
<td>4192</td>
<td>137</td>
<td>273</td>
<td>687</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>153</td>
<td>149</td>
<td>351</td>
<td>960</td>
<td>2085</td>
<td>4252</td>
<td>150</td>
<td>305</td>
<td>864</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>126</td>
<td>148</td>
<td>345</td>
<td>935</td>
<td>1859</td>
<td>4222</td>
<td>142</td>
<td>283</td>
<td>770</td>
</tr>
<tr>
<td>D</td>
<td>Best</td>
<td>239</td>
<td>484</td>
<td>1066</td>
<td>2364</td>
<td>5257</td>
<td>3417</td>
<td>227</td>
<td>868</td>
<td>2178</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>254</td>
<td>498</td>
<td>1124</td>
<td>2541</td>
<td>7800</td>
<td>14199</td>
<td>263</td>
<td>942</td>
<td>3231</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>247</td>
<td>490</td>
<td>1104</td>
<td>2421</td>
<td>6064</td>
<td>13788</td>
<td>248</td>
<td>887</td>
<td>2512</td>
</tr>
<tr>
<td>E</td>
<td>Best</td>
<td>229</td>
<td>476</td>
<td>1039</td>
<td>2227</td>
<td>5203</td>
<td>3297</td>
<td>220</td>
<td>854</td>
<td>2156</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>252</td>
<td>500</td>
<td>1105</td>
<td>2434</td>
<td>6260</td>
<td>13858</td>
<td>256</td>
<td>895</td>
<td>2594</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>243</td>
<td>491</td>
<td>1066</td>
<td>2362</td>
<td>5874</td>
<td>13587</td>
<td>237</td>
<td>868</td>
<td>2434</td>
</tr>
<tr>
<td>F</td>
<td>Best</td>
<td>222</td>
<td>461</td>
<td>1013</td>
<td>2169</td>
<td>5174</td>
<td>13176</td>
<td>216</td>
<td>839</td>
<td>2143</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>239</td>
<td>471</td>
<td>1024</td>
<td>2468</td>
<td>5759</td>
<td>13718</td>
<td>245</td>
<td>890</td>
<td>2386</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>229</td>
<td>463</td>
<td>1019</td>
<td>2361</td>
<td>5380</td>
<td>13467</td>
<td>229</td>
<td>862</td>
<td>2229</td>
</tr>
</tbody>
</table>
### 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]

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

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

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

<table>
<thead>
<tr>
<th>TEST CASES</th>
<th>SAGAP</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3786.444</td>
<td>3422.89</td>
</tr>
<tr>
<td>B</td>
<td>2836.444</td>
<td>2591.56</td>
</tr>
<tr>
<td>C</td>
<td>2795.222</td>
<td>2527.56</td>
</tr>
<tr>
<td>D</td>
<td>3585.556</td>
<td>3286.67</td>
</tr>
<tr>
<td>E</td>
<td>2616.667</td>
<td>2690.44</td>
</tr>
<tr>
<td>F</td>
<td>2559.667</td>
<td>2613.44</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>TEST CASES</th>
<th>SAGAP</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1037.11</td>
<td>231.89</td>
</tr>
<tr>
<td>B</td>
<td>990.222</td>
<td>233.78</td>
</tr>
<tr>
<td>C</td>
<td>981.111</td>
<td>241.44</td>
</tr>
<tr>
<td>D</td>
<td>3084.56</td>
<td>620.56</td>
</tr>
<tr>
<td>E</td>
<td>3018</td>
<td>626.11</td>
</tr>
<tr>
<td>F</td>
<td>2915.44</td>
<td>568.56</td>
</tr>
</tbody>
</table>

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

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

References