Task Matching and Scheduling in Heterogeneous Systems Using Simulated Evolution

Hassan Barada[1] Sadiq M. Sait[2] Naved Baig[2]

[1]Etisalat College of Engineering
Emirates Telecommunications Co.
Sharjah, UAE
email: hbarada@ece.ac.ae

[2]Computer Engineering Department
King Fahd University of Petroleum & Engineering
Dhahran, Saudi Arabia

ABSTRACT

This paper describes and analyzes the application of a simulated evolution (SE) approach to the problem of matching and scheduling of coarse-grained tasks in a heterogeneous suite of machines. The various steps of the SE algorithm are first discussed. Goodness function required by SE is designed and explained. Then experimental results applied on various types of workloads are analyzed. Workloads are characterized according to the connectivity, heterogeneity, and communication-to-cost ratio of the task graphs. The performance of SE is also compared with a genetic algorithm (GA) approach for the same problem with respect to the quality of solutions generated, and timing requirements of the algorithms.

1. Introduction

Heterogeneous Computing (HC) is the well-orchestrated use of a heterogeneous suite of machines interconnected via a high-speed network. HC is emerging as a major paradigm for scientific and high performance applications to exploit the heterogeneity in computations [1].

However, in order to run an application task efficiently in an HC environment, it is first decomposed into coarse-grained subtasks; each subtask is well suited to single machine architecture. The characteristics of subtasks and machines are then determined using *code-profiling* and *analytical benchmarking* which are both analytical steps that evaluate the matching of subtasks to individual machines [1, 2]. Each subtask is then *matched* to the most suitable machine and *scheduled* on it. Matching is defined as the assignment of subtasks to the machines in the HC system whereas scheduling comprises the ordering of execution of subtasks on each machine. The goal of efficient heterogeneous computing is then to achieve the minimal execution time of the application task when it is running in the HC system.

Task matching and scheduling in HC systems is known, in general, to be NP-complete [4]. Therefore, heuristics that attempt to reach near-optimal solutions to this problem have been proposed in the literature [3-8]. A good survey of many heuristics is in [4]. In this paper, we propose another heuristic, simulated evolution (SE), which is a powerful general iterative heuristic that has been applied to few optimization problems especially in design automation [9, 10]. This paper discusses and analyzes the implementation of this approach to task matching and scheduling in HC systems and compares its performance to the genetic algorithm (GA)-based approach [3], with respect to the quality of solutions and run time requirement of the algorithms.

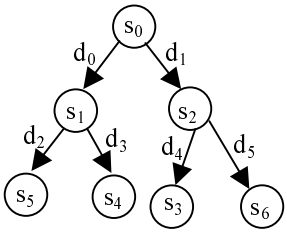
2. Problem Definition

Different HC models have been used in the literature [3-8]. The HC model used in this work is similar to the model assumed by Wang et al. in [3]. An application task is decomposed into a set of coarse-grained subtasks $Sb=\{s_i, 0\leq i<k\}$, each subtask is well suited to a single machine architecture. The data items that need to be

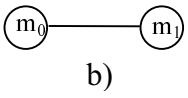
transferred between the subtasks form a set D={d_i, 0≤i<p}. An HC system consists of a set of machines M={m_i, 0≤i<l} each of which is characterized by a specific architecture such as SIMD, MIMD, special purpose FFT, etc. For the purpose of this study, it is assumed that machines are fully connected.

The estimated execution times of the subtasks in *Sb* on the machines in *M* are known a priori and are given by an $l \times k$ execution-time matrix *E*. The estimated transfer times of data items between subtasks to be transferred through the HC network are also described by an $l(l-1)/2 \times p$ transfer-time matrix *Tr*; *p* represents the number of data items while $l(l-1)/2$ represents the number of machine pairs. As an example, Figure 1a shows a DAG of 7 subtasks and 6 data items describing an application task. Figure 1b shows a graph representing a 2-machine HC system on which the application is to be executed. The estimation of execution times of the subtasks on both machines are given by the 2x7 *E* matrix shown in Figure 1c. In Figure 1d, the 1x6 matrix *Tr* gives the transfer times between m_0 and m_1 for the data items d_0 to d_5.

The task matching and scheduling problem is defined as follows: It is required to match and schedule the *k* subtasks in set *S* on the *l* machines in set *M* such that the total execution time of the application task on the HC network is minimized.



a)



b)

$$E = \begin{bmatrix} 770 & 120 & 85 & 425 & 920 & 730 & 880 \\ 790 & 730 & 60 & 375 & 910 & 650 & 800 \end{bmatrix}$$

c)

$$Tr = \begin{bmatrix} 78 & 15 & 7 & 35 & 90 & 73 \end{bmatrix}$$

d)

Figure 1. A sample HC model.

3. Simulated Evolution

Simulated Evolution is a general iterative heuristic for solving combinatorial optimization problems [9 - 11]. The book by Youssef and Sait [10] includes a good and thorough discussion of SE and its relation to other iterative.

SE starts from an initial solution. Then following an evolution-based approach, it seeks to reach better solutions from one generation to the next. The algorithm has three basic steps: *Evaluation, Selection* and *Allocation*. These steps are carried out repetitively until a stopping criterion is satisfied.

The *Evaluation* step consists of evaluating the *goodness* of each *individual* e_i of the current solution. The goodness measure is a number expressible in the range [0,1] and it is defined, for each individual e_i, as $g_i = O_i/C_i$, where O_i is an estimate of the optimal cost of individual e_i and C_i is the actual cost of e_i in its current location. O_i does not change from one generation to the next, and therefore it is computed for every individual only once during initialization.

During *selection* step, individuals of the current solution are divided into two disjoint sets; a selection set *S* and a set *R* of remaining individuals. The decision whether to include e_i in *S* or *R* is based solely on individual goodness g_i. Individuals with lower goodness values are more likely to get selected and assigned to set S. However, individuals with higher goodness values should have a non-zero probability of being selected.

Allocation alters all individuals in *S* of the current generation to produce a new generation. It involves making several trial alterations for each individual before deciding on the final configuration of the new generation. The goal of allocation is to favor improvements of quality of the current generation over the previous generation, without being too greedy. It allows the search to progressively converge to an optimal configuration where each individual is optimally allocated.

4. SE for Matching and Scheduling in HC

To implement SE, it is necessary to devise an efficient goodness measure that computes goodness value of each individual during the evaluation step. It is also necessary to devise an encoding technique that describes the state of the solution. Then, problem-dependent methods for generating an initial solution, evaluation, selection, and allocation should be designed.

4.1 Encoding Technique

We have encoded a solution to matching and scheduling in HC (MSHC) using a string of k segments where k is the number of subtasks in the DAG representing the application task. Each segment consists of two parts: a subtask identifier and a machine identifier. Pairing a subtask s_i with a machine m_j in the same segment means that s_i is assigned to machine m_j. Obviously, in this type of encoding, we have to ensure that a string represents a valid solution, which satisfies the precedence constraints in the DAG. In our encoding scheme, if a subtask s_x is placed on the left of a subtask s_y and both are assigned to the same machine m_j then s_x is executed before s_y in m_j. This encoding scheme is close to the encoding technique used by Wang et al. [3]. However, in [3], the authors used two different strings: one for matching and one for scheduling while we combine both strings in only one string for matching and scheduling.

Figure 2 illustrates the encoding scheme. The string in the figure represents a valid solution to the HC problem described in Figure 1. In Figure 2, the order of execution of subtasks in each machine is given by m_0: s_0, s_3, s_4 and m_1: s_1, s_2, s_5, s_6. This ordering means that in machine m_0, s_0 is executed first, s_3 is executed next, then s_4 last. In m_1, s_1 is executed first, s_2 second, s_5 third, then s_6 last.

s_0 m_0	s_1 m_1	s_2 m_1	s_5 m_1	s_6 m_1	s_3 m_0	s_4 m_0

Figure 2. Valid encoding string.

4.2 Generating an initial solution

To generate a valid initial solution, each subtask in the DAG is first assigned randomly to a machine in the set of machines M. Then, the DAG is topologically sorted [12]. Following the sorted order, which guarantees that data dependencies are satisfied, the subtasks are placed in successive segments. This initial valid string is then modified a random number of times as follows. A subtask s_i is selected and moved randomly to another segment in the string within its valid range of positions. The valid range of a subtask is the set of positions where the subtask can be placed without violating any data dependencies.

4.3 Evaluation

The evaluation step computes the goodness measure $g_i=O_i/C_i$ of each individual e_i of the current solution. In the SE implementation of MSHC, an individual e_i represents a subtask s_i. The *location* of an individual refers to both the pairing of s_i with m_j (the assignment of subtask s_i to machine m_j) and the position of this pair in the string (the order in which s_i is executed in machine m_j). C_i is defined as the finishing time of subtask s_i given the assignment of subtasks and the order of their execution as described in the current solution. O_i is defined as the finishing time of subtask s_i if it is placed in its *optimal* location according to a specific function F. F, in our implementation, assigns subtask s_i and all its predecessors to their best-matching machine with respect to the execution time of the subtasks on all machines.

For example, for the DAG in Figure 1, the optimal finish time of subtask s_4, according to the above function F, is its finishing time if it is assigned to machine m_1 and subtasks s_0 and s_1 are both assigned to machine m_0. In this case, O_4 = 1835 units including communication time between s_1 and s_4. This is computed before SE starts. Assuming that after SE starts, the current solution is the solution shown in Figure 2, C_4 will equal to 3123 units since s_0, s_3 and s_4 are assigned to m_0, and s_1 and s_2 are assigned to m_1.

4.4 Selection

During the selection step, at every generation of the algorithm and for each subtask s_i, a random number in the range [0,1] is generated and compared with (g_i+B), where g_i is the goodness of s_i and B is the *selection bias*. If the generated number is greater than (g_i+B), then s_i is selected and assigned to the selection set S; otherwise it is assigned to R. The selected subtasks are ordered in ascending order according to their level in the DAG, and are considered for allocation in the next step in that particular order.

The value of the selection bias B is fixed and preset at the beginning of the algorithm. It is used to have some control over the selection process as a tradeoff between implementing a fast algorithm and having a more thorough search. In our work, we have used negative values for B (between -0.1 and -0.3) for small problem sizes and positive values (between 0 and 0.1) for large problem sizes. For smaller DAGs, a negative value of B will force the selection of relatively large percentage of subtasks, hence allowing a more thorough search. In cases of large DAGs, the value of B is kept positive to restrict the number of subtasks selected in the selection step.

4.5 Allocation

The allocation step relocates all individuals in the selected set S. The strategy used in the SE algorithm for MSHC is constructive. It always chooses the best location for the subtask under consideration. First, the valid moving range of the selected subtask is determined. Next, all various combinations are tried and the schedule

lengths of corresponding solutions are computed. A combination consists of placing the subtask in a valid segment without violating the data dependency constraints and assigning it to a machine. Finally, the subtask is placed in the segment and assigned to the machine that results in the best overall schedule length. This process is tried for every subtask in the set S and the output of the allocation step is the next generation.

One parameter, which has been analyzed and studied in our SE-implementation, is a parameter we call *Y*. Y defines the number of machines to which a particular subtask can be assigned according to its execution time on all machines. For example, if Y=2, then each subtask can be assigned to two machines, its best-matching machine and its next best. This parameter limits the number of combinations tried in the allocation step and therefore controls the trade-off between the execution time of the SE algorithm and the quality of the solution generated.

5. Experimental Results

To analyze the performance of the SE-based approach, randomly generated workloads are used. Each workload corresponds to a DAG representing an application task, the number of machines in the HC system, the matrix E, and the matrix Tr. Randomly generated workloads are used since a generally accepted set of HC benchmarks does not exist [3] and it is also desirable to obtain data that demonstrates the effectiveness of the approach over a broad range of conditions. We have conducted experiments on small and large sizes.

Workloads are further classified according to their connectivity, heterogeneity and communication-to-cost ratio (CCR). The level of connectivity in a DAG defines the number of data items to be transferred between the subtasks. Heterogeneity classifies workloads according to the degree of heterogeneity of subtasks, which defines the difference in execution times of subtasks on the different machines in the HC system. CCR is defined as the ratio of size of data item over execution time of the subtask generating this item. For example, CCR = 0.1 indicates that the communication cost is low compared to the computation cost in the DAG. This describes lightly communicated subtasks. CCR=1 indicates that the communication cost is comparable to the computation cost. This describes heavily communicated subtasks.

5.1 Effectiveness of the SE algorithm for MSHC

To verify the effectiveness of the SE algorithm, one can monitor the number of selected individuals in the selection step of the algorithm as SE progresses. Initially a large number of individuals should be selected for relocation since most individuals are not optimally placed. However, in later iterations, the number of selected individuals should decrease gradually since more individuals are placed optimally and should not be relocated.

In the first set of experiments, various sizes and types of workload were generated randomly and the number of selected subtasks at every iteration of the SE algorithm was logged. Figure 3a shows the result of a sample of these experiments. The figure sketches the number of selected subtasks versus the iteration number for a workload of large size and high connectivity. The shape of the graph, which is a typical shape of all the experiments we have run, illustrates clearly that the SE algorithm for MSHC is very effective in optimally placing the tasks in their best-matching segments. Figure 3b shows the corresponding current schedule length.

5.2 Effect of Y parameter

Y represents a tradeoff between the timing requirement of the algorithm and the quality of solution to the problem. It should be expected that higher Y should generate better quality solutions. However, this parameter was studied because we anticipated that it might be dependent on the heterogeneity of workloads. It was predicted that if the application task is highly heterogeneous, a smaller Y will still give high quality solutions but with less time.

To analyze the effect of Y and heterogeneity of the workload, SE for MSHC was run on workloads of different sizes and varying levels of heterogeneity. Different values of Y (from 2 to the number of machines) were tried out and it was observed, as expected, that the timing requirements for the SE algorithm increase as Y increases. However, as far as the quality of solution is concerned, it was not as clear. For low heterogeneous workloads, increasing Y almost always improved the quality of solution. For workloads of medium and high heterogeneity, it was observed that increasing Y only improved the solution as long as Y was relatively small compared to the number of machines in the workload. Beyond a certain threshold, the quality of solutions actually started to get worse. This can be explained that in heterogeneous workloads and for large Y, many low quality solutions have to be visited before reaching good solutions. Therefore, more iterations are needed to reach a good solution. Figures 4a and 4b show typical samples of these experiments. In both experiments, the workload is of large size. Figure 4a sketches the quality of solutions for Y equals 5, 9, and 12 if the heterogeneity of the workload is low. This figure clearly shows that as Y

increases the quality of solution improves and the rate to reach good solutions also improves. However, in Figure 4b where a highly heterogeneous workload was used, the best result was for Y = 9. Increasing Y beyond 9 actually made the quality of solutions worse during the first 1000 iterations.

5.3 Comparison of SE and GA for MSHC

GA was implemented using the algorithm in [3]. We have also used workloads with different characteristics: connectivity, heterogeneity and CCR.

Figures 5, 6, and 7 represent samples of the results of the experiments that we have run to compare both heuristics. The algorithms are applied on workloads of 100 tasks and 20 machines with varying characteristics. Figure 5 shows the best schedules found by both algorithms as real time increases for a workload of high connectivity. Figure 6 shows the best schedules found by both algorithms as time increases for a workload of CCR=1. In Figure 7, the workload used is of low connectivity, low heterogeneity, and CCR = 0.1. From these samples and other experiments we have conducted, it was clear that SE produced better solutions than GA with less time, for workloads with relatively high connectivity, and/or high heterogeneity, and/or high CCR. However, as time increases SE and GA solutions were getting closer to each other. Also, as obvious from Figure 7, for low to medium connectivity, heterogeneity and CCR, the conclusion is not as clear. Many times, GA reached good solutions faster than SE as illustrated in the sample experiment shown in Figure 7.

6. Conclusion

To the best of our knowledge, simulated evolution was never applied before to the task matching and scheduling problem in heterogeneous computing environments. In this paper, we have proposed a simulated evolution-based task matching and scheduling scheme. Experiments were conducted on various types and sizes of workload to demonstrate the effectiveness of the algorithm. We have also compared our approach to the genetic algorithm approach proposed in [3]. SE performed better than GA for workloads of certain characteristics as it generates better quality solution with less time. For other workload characteristics, the difference between the two algorithms was not clear.

Acknowledgment: The authors would like to thank King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia and Emirates Telecommunications (Etisalat) Corporation, UAE, for their support.

7. References

[1] A.A.Khokar, V.K.Prasana, M.E. Shaaban, and cho-Li Wang, "Heterogeneous Computing: Challenges and Opportunities", *IEEE Computer*, June 1993, pp. 18-27.

[2] S.Chen and W.Tsai, "A Graph matching approach to optimal task assignment in Distributed Computing System using a minimax criterion", *IEEE Transactions On Computers*, C-34(3), March 1985, pp. 197-203.

[3] L.Wang, H.J.Siegel, V.P.Rowchoudhry, and A.A.Maciejewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic Algorithm-Based Approach", *Journal of Parallel and Distributed Computing*, 47(8-22), November 1997, pp. 8-22.

[4] T.D. Braun, H.J. Siegel, et al., "A Comparison Study of Static Mapping Heuristics for a Class of Meta-tasks on Heterogeneous Computing Systems", *Proceedings of Eighth Workshop on Heterogeneous Processing*, 1999.

[5] H. Topcuoglu, S. Hariri and M.Y. Wu, "Task Scheduling Algorithms for Heterogeneous Processors", *Proceedings of Eighth Workshop on Heterogeneous Processing*, 1999.

[6] A. Radulescu and A.J.C. van Gemund, "Fast and Effective Task Scheduling in Heterogeneous Systems", *Proceedings of ninth Workshop on Heterogeneous Processing*, 2000.

[7] Harmel Singh and Abdou Youssef, "Mapping and Scheduling Heterogeneous Task Graphs using Genetic Algorithms", *Proceedings of Seventh Workshop on Heterogeneous Processing*, 1997, pp. 86-97.

[8] N.S. Flan, R.F. Freund, P. Shroff, and D.W. Watson, "Genetic Simulated Annealing for Scheduling Data-Dependent Tasks in Heterogeneous Environments", *Proceedings of Seventh Heterogeneous Computing Workshop*, 1997, pp. 98-103.

[9] Ralph M. Kling and Prithviraj Banerjee, "ESP: A new Standard Cell Placement Package using Simulated Evolution", *Proceedings of 24th Design Automation Conference*, , 1987, pp. 60-66.

[10] Habib Youssef and Sadiq Sait. Iterative Algorithms and Their Applications in Engineering. IEEE press, 1999.

[11] Ly T.A. and Mowchenko J.T., "Applying Simulated Evolution to Scheduling in High Level Synthesis", *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, 1990, pp. 172-175.

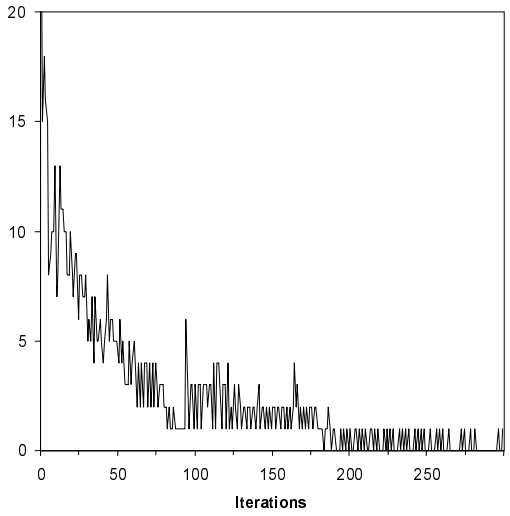
[12] Leiserson C.E., Cormen, T.H. and Rivest R.L. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1992.

Biography

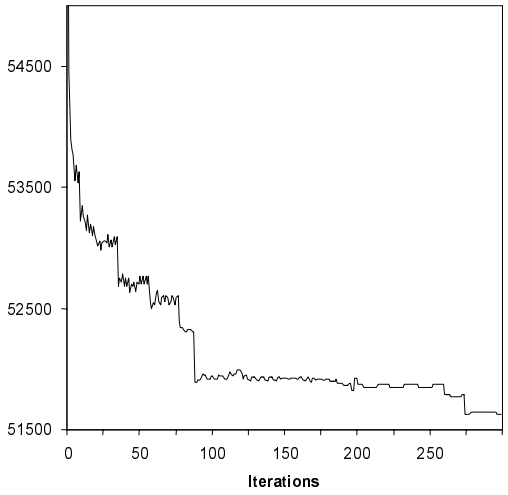
Dr. Hassan R. Barada received his BS, MS, and PhD degrees in Electrical Engineering from Louisiana State University, Baton Rouge, USA, in 1984, 1986, and 1989 respectively. From 1989 until 1995, he was an Assistant Professor in the Electrical Engineering and Computer Science department at Lehigh University, Pennsylvania, USA. From 1995 until 1999, he was an Assistant/Associate Professor in the Computer Engineering department at King Fahd University of Petroleum and Minerals, Saudi Arabia. He is currently an Associate Professor and Head of Computer Engineering department at Etisalat College of Engineering which is a division of Emirates Telecommunications Corporation, United Arab Emirates. Dr. current research interests are in the areas of parallel and distributed computing and systems, and iterative algorithms with applications to computer engineering problems.

Dr. Sadiq M. Sait obtained a Bachelor's degree in Electronics from Bangalore University in 1981, and Master's and PhD degrees in Electrical Engineering from King Fahd University of Petroleum and Minerals (KFUPM, Dhahran, in 1983 and 1987 respectively. Since 1987 he has been working at the Department of Computer Engineering where he is now a Professor. In 1981 Sait received the best Electronic Engineer award from the Institute of Electrical Engineers, Bangalore. In 1990, 1994 and 1998 he was awarded the `Distinguished Researcher Award' by KFUPM. Sait has authored over 80 research papers in International Journals and Conferences. He served on the editorial board of International Journal of Computer Aided Design between 1988-1990, and was invited to serve as a guest editor for their special issue on Hardware Description Languages. He is currently the Computer Science Editor of Arabian Journal for Science and Engineering. Sait has also served as a referee for several leading journals. He is the member of the reviewers board of the Computer-Aided Design Journal, UK. He has given invited lectures in The University of Erlangen, Nurenberg, and at IBM Research Center, Rushlikon. His current areas of interest are in Digital Design Automation, VLSI System Design, and High Level Synthesis. Sadiq M. Sait is the co-author of the book ``VLSI Physical Design Automation: Theory and Practice", published by McGraw-Hill Book Co., Europe, (also co-published by IEEE Press), December 1994; and; also the co-author of the book ``Iterative Compute Algorihtms with Applications in Engineering: Solving Combinatorial Optimization Problems", IEEE CS Press, USA, December 1999.

Naved Baig received his B.E Computer Systems Engineering degree in 1995 from NED University of Engineering and Technology, Karachi, Pakistan. He received his MS Computer Engineering degree in 1999 from King Fahd University of Petroleum and Minerals, Saudi Arabia where he is working as a Lecturer now. His research interests include iterative computer algorithms and their applications in computer engineering, computer architecture and computer networks.



a)



b)

Figure 3. Effectiveness of SE for MSHC.
a) Number of selected subtasks versus iteration.
b) Schedule length of current solution at each iteration.

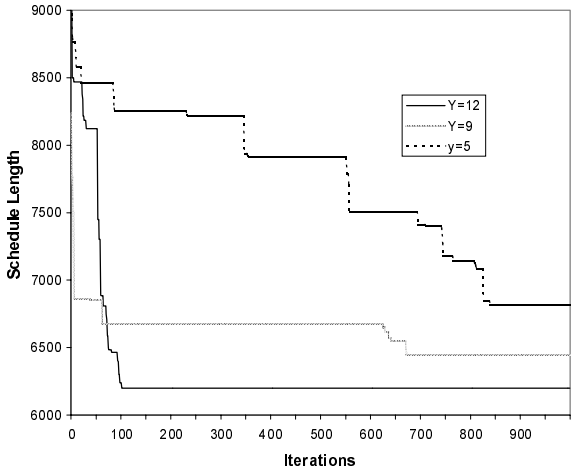


Figure 4a. Effect of Y for low heterogeneity.

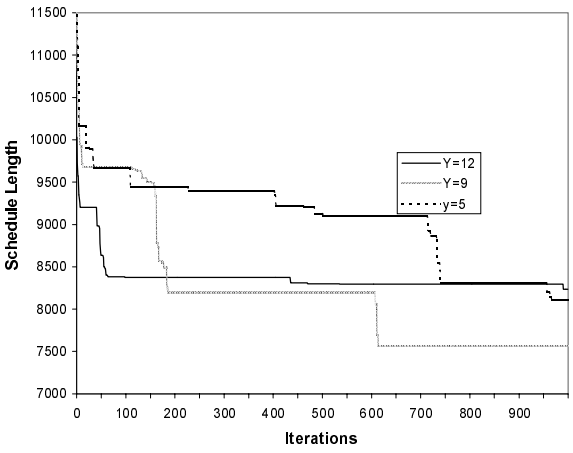


Figure 4b. Effect of Y for high heterogeneity.

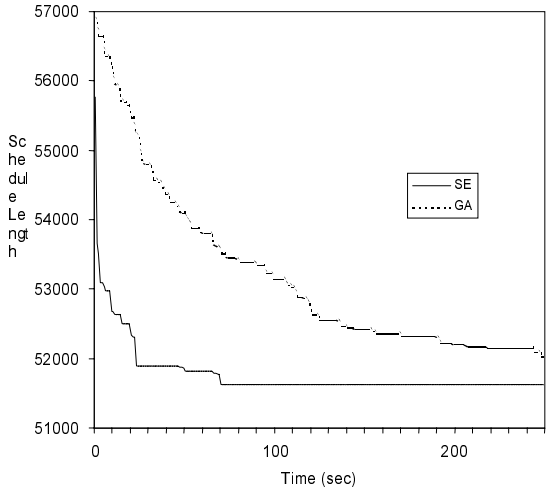


Figure 5. SE vs. GA with high connectivity.

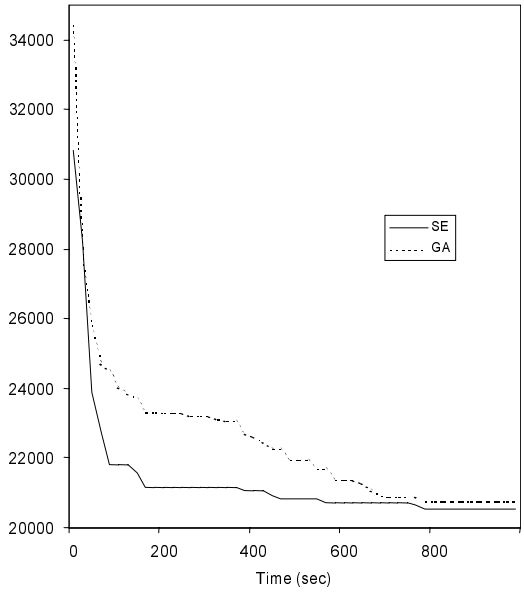


Figure 6. SE vs. GA with CCR = 1.

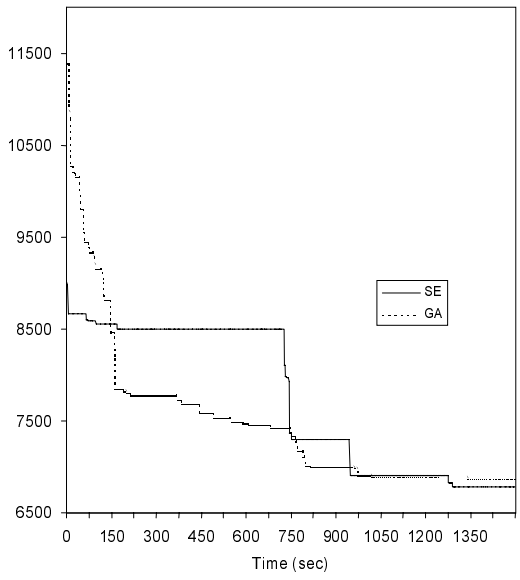


Figure 7. SE vs. GA with Low connectivity, heterogeneity and CCR.