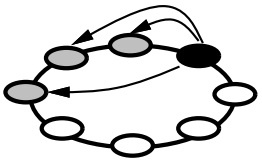

Figure 2: communication structure

algorithm	minimal cost	$\sigma(\lvert P_i \rvert)$	run time
[3]	587	0.99	78 rounds
[8]	453	0.99	78 rounds
PGA [6, 7]	430	0.00	28 min, 500 generations

Figure 3: Minimal costs found with different algorithms

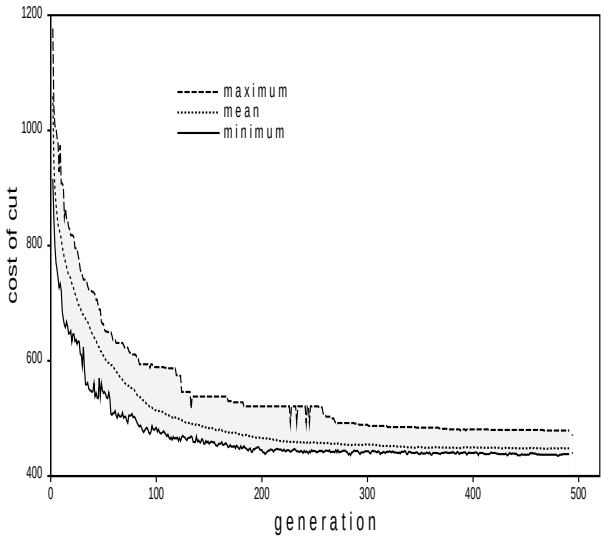

Figure 4: problem beam, 64 individuals

References

[1] G.C. Everstine. A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront. Int. J. Numer. Methods in Eng., Vol. 14, 837-853, 79.

[2] M. Gorges-Schleuter. ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy. In 3rd Int. Conf. on Genetic Algorithms, San Mateo, Morgan Kaufmann, 89.

[3] J. R. Gilbert and E. Zmijewski. A Parallel Graph Partitioning Algorithm for a Message-Passing Multiprocessor. Techn. Report 87-803, Cornell University, 87.

[4] J. H. Holland. Adaptation in natural and artificial systems. Ann Arbor, University of Michigan Press, 75.

[5] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. Technical report, Bell Syst. Techn. J., February 70.

[6] G. von Laszewski. Ein paralleler genetischer Algorithmus für das GPP. Master's thesis, Univerisität Bonn, 90.

[7] G. von Laszewski. A parallel genetic algorithm for the graph partitioning problem. In Transputer Research and Aplications 4, Proc. of the 4th Conf. of the North-American Transputers Users Group, IOS Press, Ithaca, NY, 90.

[8] D. Moore. A Round-Robin Parallel Partitioning Algorithm. Technical Report 88-916, Cornell University, Ithaca, NY, 88.

[9] H. Mühlenbein. Parallel Genetic Algorithm, Population Dynamics and Combinatorial Optimization. In 3rd Int. Conf. on Genetic Algorithms, San Mateo, Morgan Kaufmann, 89.

Using the simple string representation leads to the problem that different individuals have the same phenotype. Assume that two solutions exists which are only different in numbers for the partitions. Then a crossover operator like the one described above destroys to much information because the set of overwritten nodes becomes too large. In this case it is useful to adapt the two parent strings. The numbers of the partitions are changed in such a way that the difference between the two parent solutions is as small as possible. A more formal description can be found in [6, 7].

For larger problems, it is important to restrict the solution space. This can be done by improving the solutions with a hill climbing algorithm. Therefore, a variant of the 2-opt algorithm described in [5] is used. For all pairs of nodes, the assignment of the two nodes to a partition is exchanged, if the exchange improves the cost of the solution. This step is repeated until no improvement can be done. Instead of trying the exchange over all pairs of nodes we execute the 2-opt algorithm only on the nodes located at the border of the partitions.

The relationship between the individuals are determined by a ring. If for example, three is the size of the neighborhood, then those three individuals are in the neighborhood which lies in the ring directly behind the individual (figure 2). We also include the so far best individual in the neighborhood of an individual.

4 Results

The PGA is implemented on a 64 transputer based system. Each processor contains an individual. This implementation environment is used to experiment with the partitioning of graphs defined in [1]. There are only a few algorithms that can be compared with the PGA, because the partitioning problem is usually restricted to the bi-partitioning problem ($k = 2$). Two comparable algorithms can be found in [3, 8]. These algorithms do not use the constraint of equal partition size, so that the partitioning problem is simpler. The PGA algorithm found the best known solution for the problem instance *beam* with 918 nodes which is divided into 18 parts. Figure 3 shows the best known results found with the different algorithms. Column 3 of the table compares the standard deviation of the size of the parts found with the different algorithms. Figure 4 shows the progress of the solution. Experiments with different population sizes and neighborhood sizes shows that a small neighborhood size and a large population size should be chosen to get the best results (e.g. 4, 64) [6, 7].

5 Conclusion

The parallel genetic algorithm computes very good results for the graph partitioning problem because the search space has the property that a combination of two high valued points of the search space leads often to a higher valued point. The algorithm uses distributed selection in a restricted neighborhood. A large population size and a small neighborhood size should be chosen in order to find the best results. The definition of sophisticated genetic operators is useful if a representation is difficult to find or if the choosen representation does not take advantage of the structure of the problem.

3 Representation, Operators, Population Structure

To apply the parallel genetic algorithm to the GPP, a representation of solutions has to be defined. We chose the simple string representation described in section 2.
If a crossover operator destroys too much information gained in previous steps, the genetic algorithm degenerates to a simple random search algorithm. To take advantage of the information already gained, a structural crossover operator is defined. It copies whole partitions from one solution into another.

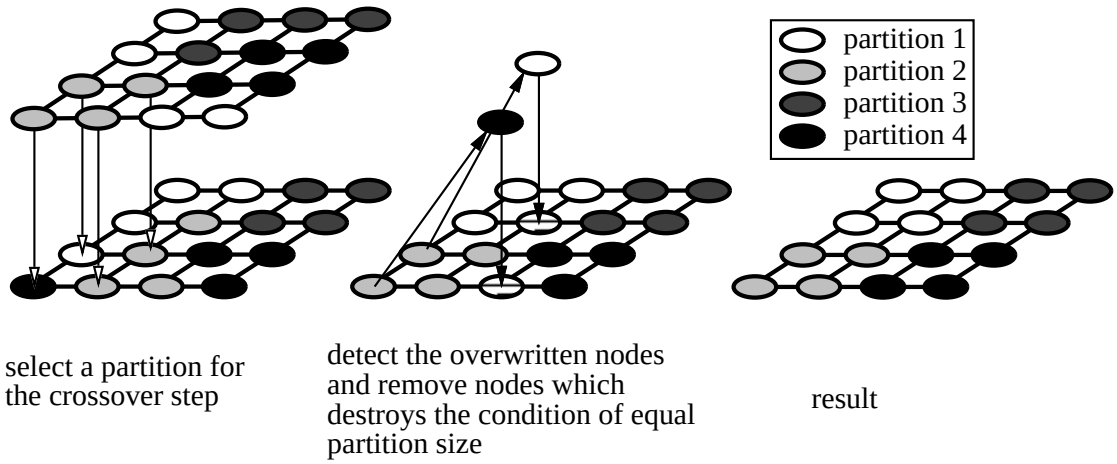

Figure 1: Recombination of two solutions

Figure 1 depicts the recombination of two solutions. A grid with 4×4 nodes is to be divided into 4 partitions. To show the recombination step more clearly, colors are used in the figure instead of numbers for the partitions.
First, a partition is randomly chosen in a parent solution (the light gray partition). Then this partition is copied into the other parent solution. Because this copy process may destroy the constraint of equal sizes of the partitions, a repairing operator is applied. In the repairing step, all nodes in the temporary solution which are not elements of the copied partition, but have the same color as this partition, are detected.
These nodes are marked in the second part of the figure 1 with horizontal lines. To assign these nodes to a partition, they have to be marked (e.g. randomly) with the colors of those nodes which have been overwritten by the copied partition. In the example the white and the black partitions have one node too few. So the nodes marked with horizontal lines are relabeled with the color white and black.

Mutation is applied after the recombination step. To avoid creating invalid solutions, mutation is defined as a number of exchanges of two numbers in the coding. The exchange step is repeated as long as the difference between the new descendant and one parent is under a specific limit. The difference is defined as follows:

$$\text{difference } (a_1 ... a_n, b_1 ... b_n) = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } a_i \neq b_i \\ 0 & \text{otherwise} \end{cases}$$

If we represent the solutions of the problem as a fitness landscape in a certain configuration space, we see that a PGA tries to jump from two local minima to a third minimum, using the crossover operator. This jump is (probabilistically) successful, if the fitness landscape has a certain correlation. The PGA can be applied to other combinatorial problems with great success [9, 2]. The parallel genetic algorithm can be described as follows:

Specification: A genetic representation of the optimization problem has to be defined.

Initialization: An initial population and its population structure has to be created. Local hill climbing is done independently by each individual to increase its fitness.

Local selection: A partner for mating is selected by each individual in its *neighborhood.*

Recombination: A new descendant is created with genetic operators using the codings of the parents.

Hill climbing: Local hill climbing is done to increase the fitness of the descendant.

Replacement: The parent is replaced, if the fitness of the improved descendant is, e.g., at least as good as the worst in the local neighborhood. If not finished, go to **Local selection.**

2 The Graph Partitioning Problem

The k way graph partitioning problem is a fundamental combinatorial problem which has applications in many areas of computer science (e.g., design of electrical circuits, mapping) [5]. Mathematically we can formulate the GPP as follows:
Let $G = (V, E, w)$ be an undirected graph, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges and $w : E \mapsto I\!N$ defines the weights of the edges. The GPP is to divide the graph into k disjunct node subsets $P_1 ... P_k$, such that the sum of the weights of edges between the subsets is minimal, and the sizes of the subsets are nearly equal. The subsets are called *partitions.* The edges between the partitions are called *cut.* Let $P_1, ..., P_k$ be the partitions. Then the string $(g_1 g_2 ... g_n)$ describes the partition:

$$v_i \in P_a \iff g_i = a \quad a \in \{1, ..., k\} \quad \forall i \in \{1, ..., n\} \quad .$$

Node v_i is assigned to the partition with the number g_i. Instead of minimizing the cost of the cut we maximize the sum over all weights of edges between nodes in the same partitions. This is an equivalent problem because the total cost of edges is constant.

$$c(g_1 g_2 ... g_n) = \sum_{\substack{1 \le i < j \le n \\ g_i = g_j}} w(v_i, v_j) \quad .$$

The advantage of this cost function is that a selection operator can be easily formulated. The following PGA does not change the sizes of the subsets during the computation.

Partitioning a Graph with a Parallel Genetic Algorithm

Gregor von Laszewski
gregor@cis.ohio-state.edu
The Ohio-State University, CIS
2036 Neil Avenue
Columbus, Ohio 43210-1227

Heinz Mühlenbein
muehlen@gmdzi.uucp
Gesellschaft für Mathematik
und Datenverarbeitung
W-5205 St.Augustin, Germany

Abstract

We present a parallel genetic algorithm for the k way graph partitioning problem. The algorithm uses selection in local neighborhood and sophisticated genetic operators. For a sample problem the algorithm has found better solutions than those found by recent GPP algorithms. The success of the parallel genetic algorithm depends on the representation, a suitable crossover operator and an efficient local hill climbing method which is used to restrict the solution space.

1 Parallel Genetic Algorithms

The basic idea of genetic algorithms [4] is to start parallel search with a population of N *individuals* represented by *chromosomes*. In every generation the fitness of each individual is evaluated. Parents are selected according to their fitness, so that better individuals are more often selected for mating. Descendants are produced by combining the chromosomes of their parents. The exchange and variation of genetic material is controlled by *genetic operators* such as *crossover* and *mutation*.

Parallel genetic algorithms (PGA) use three major modifications compared to genetic algorithms. First, the individuals are living in a 2-D world with restricted neighborhood relations. Therefore, the selection of a mate is done by each individual independently in its neighborhood. Second, each individual may improve its fitness during its lifetime e.g., by local hill climbing. Third, an individual replace its parent only if it fulfills a specific condition.

The PGA runs asynchronously with maximal efficiency on MIMD parallel computers due to the fact that the individuals are active and not acted on as in genetic algorithms.

The success of the PGA depends on the genetic representation of the combinatorial problem, a suitable crossover operator and an efficient local hill climbing method.

We will show the power of the PGA with an application of the k way graph partitioning problem (GPP). The PGA has computed solutions for very large problems, which are comparable to or even better than any other solution found by other heuristics.