Preserving Locality for Optimal Parallelism in Task Allocation

A. Schoneveld, J.F. de Ronde, P.M.A. Sloot

Parallel Scientific Computing and Simulation Group
Faculty of Mathematics, Computer Science, Physics and Astronomy
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
The Netherlands, http://www.wins.uva.nl/research/pscs/
e-mail:{arjen,janr,peterslo}@wins.uva.nl

Abstract. Genetic Algorithms have been applied to several combinatorial optimisation problems, including the well known task allocation problem, originating from parallel computing. We introduce random task graphs as a model of applications which display irregular global communication patterns. Uniform crossover is the standard genetic recombination operator, that is applied to solution encoded chromosomes. However, application of a uniform crossover may heavily disrupt low cost sub-solutions, or building blocks, of a chromosome. Therefore, we define a locality preserving recombination operator, exploiting the connectivity of the task graph. Experiments show that this new operator increases the convergence rate of the Genetic Algorithm applied to the task allocation problem.

1 Introduction

An essential problem in the field of parallel computing is the so called *Task Allocation Problem*(TAP). Given a set of parallel communicating tasks (a parallel application) and a parallel distributed memory machine, find the optimal allocation of tasks onto the parallel system. It is well known that the TAP is an NP-hard optimisation problem [2]. The class of NP-hard problems is known to be solvable in a sub optimal way using non deterministic or heuristic algorithms like Genetic Algorithms (GA) [8] or Simulated Annealing (SA) [10].

In previous work we have described in detail, a tool which implements an optimisation kernel embedding both a Genetic Algorithm and Simulated Annealing [5]. We have also introduced the method of *Redundant Decomposition* [3][4], proposing to first apply a domain decomposition prior to the actual allocation or mapping procedure. In extension to the sequential Genetic Algorithm, a parallel Cellular Genetic Algorithm has been studied in [12]. The application domain of this previous work was limited to computational meshes, used in parallel finite element applications. Furthermore, a standard genetic recombination operator, uniform cross-over, has been applied in constructing successive generations.

Encoding of an arbitrary task graph as a one dimensional chromosome, can be easily shown to heavily disrupt the exploitation of low cost sub allocations. Therefore, we have defined a new crossover operator, *cluster crossover*, which is sensitive to optimally allocated allocations of parts of the task graph. In this work, we consider a more generic application model than that of finite ele meshes, namely random task graphs with adjustable connectivity [6]. With purpose of studying the convergence behavior of uniform cross-over compa to this new, structured, recombination operator.

This paper is structured as follows. In section 2 both the static parallel plication as well as the machine models, used in this work, are presented. Al the objective function, quantifying the costs of an allocation is formulated section 3, a general framework for evolutionary strategies, an *Abstract Cellu Genetic Algorithm* (ACGA), is given. A *Parallel Cellular Genetic Algorithm* presented as an instance of this ACGA. At the end of this section, the clu crossover operator is defined. Section 4 presents some experimental results, re resentative of the performance of this new structured recombination opera for a variety of random task graphs. Finally, we will summarise and discuss findings in sections 5 and 6.

2 The Task Allocation Problem

In order to facilitate a study on abstract parallel applications a random gra representation as a model of static communicating parallel tasks is introduce Each task is assigned a workload and every pair of tasks (vertices) in the ta graph is connected with a probability γ. A message size is assigned to each li between two communicating tasks. Work loads and message sizes are restricte to be constant. Furthermore the target processor topology is assumed to a static parallel machine that is fully connected and homogeneous. That communication channels between all processor pairs are bi-directional and ha equal bandwidths. Moreover, the processors are homogeneous, i.e. they ha identical constant performance.

The metric for deciding on the quality of a task allocation is the turn-arou or execution time. A variety of cost models that are based on a graph repr sentation can be found in literature. For example, the following cost functio equation (1) [9], is known to model the actual execution time for a given ta allocation with reasonable accuracy. Of course it is a simplification of the re situation, i.e. message latencies and network congestion are neglected.

$$H = \max_{q \in \mathcal{Q}} \left(\sum_{u_i \in \mathcal{U}^q} W_{u_i}/S_q + \max_{u_i \in \mathcal{U}^q, u_j \in \mathcal{A}(u_i)} W_{u_i u_j}/S_{pq} \right), \quad (1$$

where

- u_i is a task in the parallel task graph
- $\mathcal{Q}$: the set of processors
- $\mathcal{A}(u_i)$: the set of tasks connected to task u_i
- $\mathcal{U}^q$: set of tasks u_i residing on processor q
- W_{u_i}: Work associated with task u_i (e.g. in terms of flop)
- S_q: is the processor speed for processor q (e.g. in flop/s)
- $W_{u_i u_j}$: Number of bytes to be sent, due to nodal connectivity, between ho processor of task u_i and task u_j.
- S_{pq}: is the bandwidth of the route between processor p and q (in bytes/

A property of this specific function is that it assumes that the execution time is determined by the "slowest" processor in the parallel machine.

Because the value of H (equation (1)) can only change in task transfers that involve the slowest processor, it is not very sensitive to task rearrangements. The corresponding search space is highly discontinuous and consists of many plateaus, which makes it difficult for many local search methods to explore. A cost function which still captures the essential characteristics of optimal allocations, but which defines a smoother search space, is equation (2) (see e.g., [11]).

$$H = \beta \sum_{p} W_p^2 + (1-\beta) \sum_{p>q} C_{pq}, \tag{2}$$

where

- $W_p = A_p/S_p$, with A_p: $\sum_{u_i \in \mathcal{U}^p} W_{u_i}$, total work on processor p in terms of flop.
- $C_{pq} = M_{pq}/S_{pq}$, with M_{pq}: $\sum_{u_i \in p, u_j \in q} W_{u_i u_j}$.
- β is a control parameter, corresponding to the communication/calculation ratio (τ_{com}/τ_{calc}) [7].

Equation (2) has the *locality property*, which means that incremental changes in a task allocation can be propagated into the cost without having to recalculate the entire cost function, which is the case for equation (1) [11]. This is specifically useful if an optimisation algorithm is applied that is based on incremental changes (e.g. SA), which can exploit the calculation of cost differences. A disadvantage of using equation (2) is the fact that it is a not a correct model for the absolute cost. The objective is to minimise the variance in the workload distribution simultaneous with the communication volume of the whole system, as opposed to optimisation of the execution time of the slowest processor in equation (1).

3 Parallel Cellular Genetic Algorithm

In [1] a generic algorithm, the so-called Abstract Genetic Algorithm (AGA), for both SA and GA was introduced. However, the AGA was not designed to facilitate parallelisation. To avoid the use of global information which is necessary in the AGA, we introduce a local spatial neighbourhood structure. The main idea behind the proposed ACGA is to make an analogy between the chromosome (or solution vector) and a cell in Cellular Automata. Each chromosome is assigned to a cell, which explicitly defines its neighbourhood structure. All communication is local and cells can only interact with direct neighbours. Consequently we can formulate the ACGA:

```
Initialise
DO
  FOR EACH cell in the population
     DO
        Choose a parent list (choice)
        Recombine parent list (production)
        Mutate the offspring
```

```
      Evaluate offspring
    IF offspring meets some criterion (selection)
    THEN
      accept offspring
    ELSE
      leave the current chromosome in its place
    ENDIF
  ENDFOR
UNTIL maximum number of generations (iterations)
```

From the ACGA pseudo-code above a parallel CGA with local selection ca derived straightforwardly. We only have to select the various genetic operat First the selection operator. A conventional GA uses a global method to se the parents. One example is roulette wheel selection. With a CGA the par are selected from a neighbourhood of size $(2r+1)^2$, where r is the radius, usi fitness function F. A cell is chosen as a parent by picking a uniformly distrib random number $\xi \in [0,1)$ which satisfies the following rule:

$$\xi < \frac{F(x_m)}{\sum_{x_j \in A_{k(r)}} F(x_j)}.$$

where $A_{k(r)}$ is the neighbourhood with radius r of cell x_k, including x_k, a $x_m \in A_{k(r)}$.
We call this selection mechanism Local Roulette Wheel (LRW).

Another possibility is tournament selection, which we identify by Local To nament Selection (LTS) in the case of CGA. There is an advantage in using L over LRW in small neighbourhoods, because LRW suffers from sampling err when used on small populations [8].

As a recombination operator we can take the unchanged GA-crossover op ator, because it is already a local operation. Also the GA-mutation operator c be used. Crossover between two cells is done by taking a uniformly distribu random number $\psi \in [0,$ *length (chromosome)*$]$ as the splitting location. Mutati is realised e.g. by "bit-flipping" every bit of a chromosome with a probabil p_m. Where a bit is an n-ary number ($n \geq 2$). Evaluation means calculating t fitness of the new chromosome. Because only one child out of the two cre can be accepted, a child selection criterion must be applied. In the experim we only accepted the fittest child. Note that a maximum radius for the CGA identical to a GA with global selection. In previous work we studied the influe of the radius size on the selection pressure [12]. It was shown that LTS does n deviate from global selection tournament selection for $r > 1$.

Another instance of the ACGA is a special variant of Simulated Anneali Cellular Simulated Annealing (CSA). To introduce locality in the SA algorit we use an alternative approach where several configurations exist together o 2-dimensional (2D) grid. These configurations only know of the existence of ot configurations in their direct neighbourhood. This neighbourhood is defined a localised spatial structure on the grid. If a new configuration has to be evalu for acceptance, not only the previous configuration is taken into account, b also its neighbourhood set. Rejection of a new configuration can cause any of the configurations in a neighbourhood set to take over the current spatial grid location. Experiments with CSA for the Travelling Salesperson Problem were reported elsewhere [13].

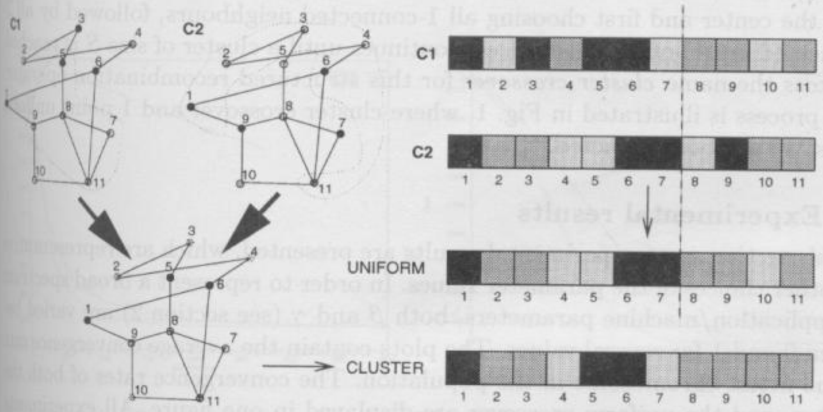

Fig. 1. *Schematic representation of the structured crossover vs. the uniform operator. Two task allocations, C1 and C2, are selected as children for a recombination operation. The left hand side of the figure depicts the cluster crossover: a sub cluster (dotted line) is chosen in C1, which is recombined with the complementary cluster in C2. The right hand side of the figure depicts the uniform crossover: an arbitrary crossover point (dotted line) is chosen and the two sections of C1 and C2 are combined. The grey-values in the chromosomes represent the processor to which a task is allocated.*

An important aspect of Genetic Algorithms is the exploitation of so called building blocks or schemata [8]. Building blocks represent partially known gene values, which are most significant for the associated cost of a complete allocation. Recombination operators are applied to mix optimal building blocks from chromosomes in the current population. An important property, for efficient sampling of optimal building blocks, is their compactness. This means that the defining length in the chromosome must be short enough, such that destruction of building blocks by uniform crossover is minimised. For graph problems, the defining length of the schemata can often be very long, because the inherent locality of the graph can not be represented by a one dimensional encoding. However, the disruption process can be avoided if the internal structure of the problem domain is used as additional knowledge. Two approaches can be taken in order to reach this goal. Either the coding of solutions must somehow represent the inherent structure of the problem domain or the genetic operators must be able to exploit this. In this work we take the latter approach for reasons of generality, i.e. the same coding must be used for different optimisation methods. The internal structure in a random task graph is obvious, namely the connectivity between the vertices. In order to use this information, we propose a recombination method, which exploits *connected sub graphs* instead of *sub chromosomes*. The procedure is as follows: First choose a random vertex ν and a random size $S \in [0, n]$, where n corresponds to the number of vertices. Subsequently, a cluster of S connected vertices is chosen. The cluster is constructed taking vertex ν as the center and first choosing all 1-connected neighbours, followed by all 2-connected neighbours. This process continues until a cluster of size S is reached. We toss the name *cluster crossover* for this structured recombination operator. The process is illustrated in Fig. 1, where cluster crossover and 1 point uniform crossover are both depicted.

4 Experimental results

In this section some experimental results are presented, which are representative for other choices of the parameter values. In order to represent a broad spectrum of application/machine parameters, both β and γ (see section 2) are varied between 0 and 1 for several values. The plots contain the average convergence rate of the fittest chromosome in the population. The convergence rates of both the cluster- and the uniform crossover are displayed in one figure. All experiments have been performed with a PCGA with $r = 2$, a crossover probability of 0.9, and a mutation probability of $1/n$, where n is the number of tasks. A population is considered converged whenever the cost value of the best individual has not changed over 100 generations. For each parameter instance, the experiments have been averaged over 25 CGA runs. All comparisons where done on the same instance of a specific random task graph (parametrised by n and γ). The initial populations of the different runs were generated randomly, which cause for cost discrepancies in the first generation. The CGAs have been executed on a 16-node partition of a Parsytec CC-40 machine.

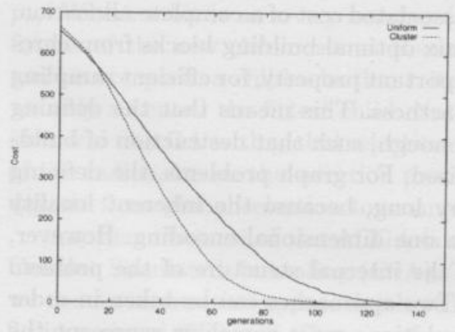

Fig. 2. $n = 64, P = 8, \gamma = 0.2, \beta = 0.0$

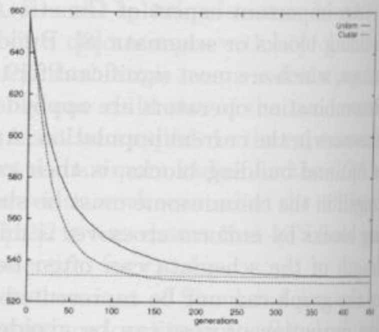

Fig. 3. $n = 64, P = 8, \gamma = 0.2, \beta = 0.15$. *The horizontal dotted line indicates the lowest average cost of the uniform crossover.*

An experiment has been performed in which only the communication term is enabled (see Fig. 2). The optimum of this specific case is known, and corresponds to a sequential allocation with cost value zero. For two higher values of β, experimental data are displayed in Figs. 3 and 4. In Fig. 3 a horizontal line is displayed, indicating the lowest average cost value in the uniform crossover convergence plot.

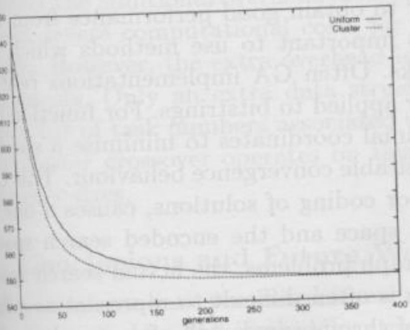

Fig. 4. $n = 64, P = 8, \gamma = 0.2, \beta = 0.30$

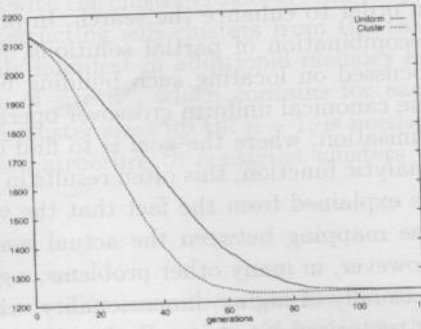

Fig. 5. $n = 64, P = 8, \gamma = 0.8, \beta = 0.30$

In Fig. 5, the convergence behaviour on a highly connected task graph ($\gamma =$ 0.8) is shown, with the same β as in the previous figure (4). Next we consider the convergence on larger task graphs. The number of tasks is taken to be 128, still using a 8-node fully connected target topology. Fig. 6 displays the results for a moderately connected graph ($\gamma = 0.1$) with $n = 128$ and $P = 8$. For an even larger task graph ($n = 200$), more processors ($P = 16$), 25($\gamma = 0.25$) and $\beta = 0.17$, the average convergence is depicted in Fig. 7. Note that we have only displayed the data up to 1000 generations.

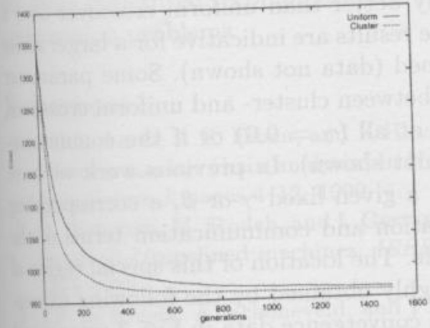

Fig. 6.
$n = 128, P = 8, \gamma = 0.1, \beta = 0.06$

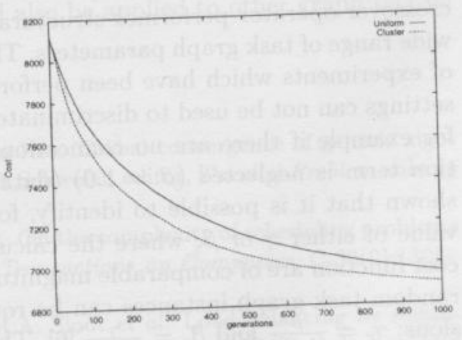

Fig. 7.
$n = 200, P = 16, \gamma = 0.25, \beta = 0.17$

5 Discussion

Genetic Algorithms have gained increased attention over the last few years as an optimisation method for so called hard combinatorial problems. The strength of this evolutionary technique is twofold. On the one hand, the selection scheme in combination with heuristic search induces a search mechanism which is less sensitive for getting stuck in local minima than deterministic methods. On the other hand, a crossover mechanism exploits low cost solutions of sub problems in order to enhance the search. In order to obtain good performance from this recombination of partial solutions, it is important to use methods which are focussed on locating such building blocks. Often GA implementations rely on the canonical uniform crossover operator applied to bitstrings. For function optimisation, where the goal is to find optimal coordinates to minimise a specific analytic function, this often results in desirable convergence behaviour. This can be explained from the fact that the vector coding of solutions, causes a one to one mapping between the actual search space and the encoded search space. However, in many other problems, e.g. graph problems, the actual search space is usually of higher dimensionality, which is often difficult to represent as a flat or vectorised bit string. Recombination of chromosomes will often result in the mixing of two completely incompatible solutions. If instead the knowledge of the connectivity in the graph is added to a graph specific recombination operator, it is possible to use optimal building blocks in the mixing process. Due to the fact that the communication term in the TAP cost function enforces locality of neighbouring tasks, optimal partial solutions can be located by a connectivity sensitive operator. In this paper, we have proposed exactly such a structured operator: cluster crossover. For reasons of generality, random task graphs have been used as problem instances. Although random task graphs aim on applications with a global communication pattern and as such are not entirely general, they still cover a broad range of scientific applications where long range interactions are present (e.g. in n-body simulations with Coulomb interactions).

The experimental results presented in section 4 indicate that the cluster crossover operator performes structurally better than uniform crossover over a wide range of task graph parameters. The results are indicative for a larger series of experiments which have been performed (data not shown). Some parameter settings can not be used to discriminate between cluster- and uniform crossover, for example if there are no connections at all ($\gamma = 0.0$) or if the communication term is neglected ($\beta = 1.0$) (data not shown). In previous work we have shown that it is possible to identify, for a given fixed γ or β, a corresponding value of either γ or β, where the calculation and communication terms of the cost function are of comparable magnitude. The location of this special region of random task graph instances can be roughly obtained by the following expressions: $\gamma_c = \frac{\beta}{(1-\beta)}$ and $\beta_c = \frac{\gamma}{(\gamma+1)}$ [6]. The convergence data in Fig. 3 correspond to such a specific β value for $\gamma = 0.2$. Compared to β values outside this region, the difference between cluster and uniform crossover is most pronounced and therefore most difficult to optimize. The ratio of the convergence steps to reach the same cost value between uniform and cluster crossover is approximately 3.5

5. J.F. de Ronde, A. Schoneveld, and P.M.A. Sloot. A genetic algorithm based tool for the mapping problem. In E.J.H. Kerckhoffs, P.M.A. Sloot, J.F.M. Tonino, and A.M. Vossepoel, editors, *ASCI'96: Proceedings of the 2nd annual conference of the Advanced School for Computing and Imaging*, pages 174–179, 1996.
6. J.F. de Ronde, A. Schoneveld, and P.M.A. Sloot. Properties of the task allocation problem. Technical Report TR-CS-96-03, University of Amsterdam, 1996.
7. G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors*, volume 1. Prentice-Hall, 1988.
8. D.E. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
9. J. De Keyser and D. Roose. Load balancing data parallel programs on distributed memory computers. *Parallel Computing*, 19:1199–1219, 1993.
10. S. Kirkpatrick, C.D. Gelatt jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
11. N. Mansour and G. Fox. Allocating data to multicomputer nodes by physical optimization algorithms for loosely synchronous computations. *Concurrency: practice and experience*, 4(7):557–574, 1992.
12. A. Schoneveld, J.F. de Ronde, P.M.A. Sloot, and J.A. Kaandorp. A parallel cellular genetic algorithm used in finite element simulation. In H-.M. Voigt, W. Ebeling, I. Rechenberg, and H-.P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN IV)*, pages 533–542, 1996.
13. P.M.A. Sloot, J.A. Kaandorp, and A. Schoneveld. Dynamic complex systems (dcs): A new approach to parallel computing in computational physics. Technical Report TR-CS-95-08, University of Amsterdam, 1995.

for this specific instance. Though less dramatic, outside this special region, the average convergence behaviour of the cluster crossover is also superior compared to its uniform variant. For Figs. 6 and 7, the β parameters have been chosen such that they are near this special region. Also for large task graphs ($n = 128$ and $n = 200$) and more processors ($P = 16$) the cluster crossover increases the convergence of the average cost of the best individual.

Besides the algorithmical convergence aspects, it is of course important to consider the additional overhead associated with the cluster crossover. One would expect severe computational costs for constructing sub clusters from the task graphs. However, the extra overhead is only manifest in additional memory requirements. Only an extra data structure of size n^2, which contains for each task a list of task numbers associated with cluster sizes up till $n - 1$, is needed. The cluster crossover operates on this data structure to construct clusters of arbitrary sizes.

6 Conclusions and Future Work

In this work we have presented a new structured crossover operator for GA that can be applied to the Task Allocation Problem. It has been shown that this novel genetic operator outperforms the standard uniform crossover convergence rate in a broad spectrum of task graphs. The initial convergence rate of the cluster crossover is faster than that of the uniform crossover operator. Given the fact that the computational overhead of the cluster crossover is negligible, it can be applied as a fast method for obtaining sub-optimal solutions to the task allocation problem. In the future we intend to loosen restrictions on the TAP instances, which are still present in this current study. The target parallel machine topology, for example, has been constrained to fully connected communication networks. Also, the task graphs are defined to model parallel applications with global communication patterns. Hence, we will study the effect on both arbitrary machine topologies and task graphs with communication ranges varying from maximally local to maximally global. It would be interesting to know whether the same structured recombination could also be applied to other graph related optimization problems.

References

1. E.H.L. Aarts, A.E. Eiben, and K.H. van Hee. Global convergence of genetic algorithms: a markov chain analysis. In H.P. Schwefel, editor, *Parallel Problem solving from Nature I*, pages 4–12, 1990.
2. D. Bernstein, M. Rodeh, and I. Gertner. On the complexity of scheduling problems for parallel/pipelined machines. *IEEE Transactions on Computers*, C-38(9):1308–1313, 1989.
3. J.F. de Ronde, A. Schoneveld, and P.M.A. Sloot *et al.* Load balancing by redundant decomposition and mapping. In H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, editors, *High Performance Computing and Networking (HPCN'96)*, pages 555–561, 1996.
4. J.F. de Ronde, A. Schoneveld, and P.M.A. Sloot. Map: a tool for load balancing by redundant decomposition and mapping. To appear in: FGCS Special issue HPCN'96.