How to Divide a Catchment to Conquer its Parallel Processing
An Efficient Algorithm for the Partitioning of Water Catchments

Olaf David, Michael Grübsch
Friedrich-Schiller-University Jena
Institute for Geographic Sciences, Department of Geoinformatics, Hydrology and Modelling
Löbdergraben 32, 07743 Jena, Germany
c5olda@geogr.uni-jena.de, mmg@geogr.uni-jena.de

Abstract: For the sake of distributed modelling, water catchments are subdivided into a set of subareas whose number is theoretically unbounded. To cope with such a computational challenge – 100 to 100,000 subareas are usual – the subareas which are to be calculated separately could be distributed among several processors respectively network computers. By the means of graph theory and the analysis of 'typical' routing systems, the present paper develops an heuristic algorithm to partition the set of subareas into subsets each of them assigned to a particular processor such that the computational load of the processors is balanced and the need for inter-process-communication is minimal.

1. INTRODUCTION: WATER CATCHMENTS, PARALLEL PROCESSING AND GRAPH THEORY

Modelling and simulation in Hydrology supports the environmental planning process in a wide range of application. In regional water catchment management the reliable prediction of water resource budget based on an integrated system analysis and their modelling results drives management decisions for landscape composition. The simulation of hydrological processes deals with prognostic analysis of 'what-if' scenarios for delineation of reproducible criteria for evaluation of efficient resource management.

A milestone in development of river catchment modelling was the Stanford Watershed Model [CRAWFORD et al., 1966]. Since this time the computing environment has changed rapidly, the resolvable problem complexity in terms of high granular definition and usage of data respectively processes within a catchment (depending on the spatial and time scale) could be increased due to modern techniques in high performance computing. This growth was crucial to the genesis of simulation in general [LAU, 1996; SINGH, 1995]. Beside the application of cost intensive architectures (massive parallel systems) the standardisation of software libraries and languages and environments such as DCE, PVM, MPI, Fortran 90 or HPF offers wide suitable platforms also for hydrological problem solution.

Using JAVA™ architecture for hydrological simulation a methodical (object-oriented Threads classes) and technical (transparent heterogeneous Internet migration) impact to the solution of large application could be attained.

For the sake of distributed hydrological modelling a water catchment is classified or subdivided by various criteria. This yields a patchwork of subareas with several classification specific and individual properties. The approach of distributed modelling takes this individuality into account and models the whole catchment by modelling and computing the subareas one by one.

Although the subareas are calculated separately they are related to each other by the water flowing through the catchment. On the patchwork of areas a network of routing channels is defined, which determines how the water (groundwater, interflow and surface) runs through the catchment. By this way one can examine source, amount, portion and chemical composition of the water flowing through the areas. This hydro-dynamical routing network fixes the order of computation by defining the sources from where each area requires its input that is to calculate before the area itself can be computed (fig 1).

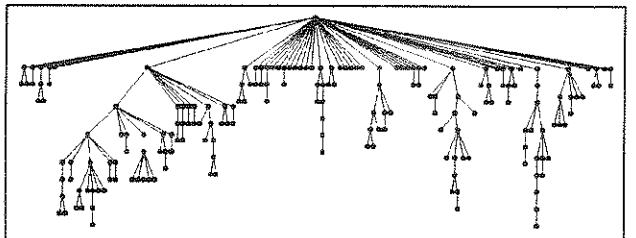

Figure 1: Routing graph of a simple catchment[1]

Depending on the degree of the classification (that means the number of independent criteria) and the extent of the catchment the number of different subareas is theoretically unbounded. However 100 to 100,000 subareas are usual in practice and can be computationally managed with a reasonable amount of time. Nevertheless 1,000,000 and more areas may be possible in some cases.

One way to cope with such a computational challenge is to 'divide and conquer': by distributing the areas of a catchment onto several processors respectively local computers one can reduce the amount of time to compute the whole routing network rapidly. Today this idea is easy to realise for there are more and more multi-processor-

[1] The Data visualized in the given figures originated from the catchment considered in [STAUDENRAUSCH, 1997].

machines available on the desktop market and thanks to the recently invented programming language JAVA™ which allows platform-independent parallel-processing in heterogeneous networks.

Due to the various inter-connections between the areas dividing is not a straight-forward-solvable problem. An efficient partitioning of catchment areas requires i) that the areas are evenly distributed among the processors depending on the power of the processors and the area's internal computational complexity, and ii) that there are as few as possible hydro-dynamical dependencies between areas assigned to distinct processors to keep down the need of inter-process communication (data exchange between areas).

By abstracting from the context a catchment and its routing network is regarded in terms of graph theory: areas are represented by nodes and the relation of drainage from one area to another is reflected by an edge between the corresponding nodes; the whole network is a directed acyclic graph. By this way the given problem of finding sets of areas is transformed into the task to find a set of disjoint subgraphs such that all of the subgraphs contain almost the same number of nodes and the number of edges connecting two different subgraphs is minimal.

Although a systematic solution of this assignment-problem would be possible, it is impossible in practice. The so called graph-partitioning-problem is known to be NP-complete. That means that its computational complexity increases exponentially with the number of subareas. Thus the typical topographic structure of water catchments is to be analysed and considered while developing a reasonable heuristic to find a good partition of a given catchment.

The present paper

1. formulates the problem in a general and mathematical (graph theoretical) way and develops guidelines to evaluate the quality of a given graph partitioning,
2. analyses and classifies the structure of water catchments in the terminology of graph theory,
3. invents an algorithm, which calculates the required partitioning in a reasonable amount of time and with a sufficient optimised result, and
4. estimates this algorithm with respect to its software-technical advantages and disadvantages and the quality of its results.

2. GRAPH THEORY DEFINITIONS

A *directed graph* $G = (N, E)$ consists of a set N of nodes and a binary relation $E \subseteq N \times N$ of edges between these nodes. For $(x, y) \in E$ the node x is called the *immediate predecessor* of y, and y is called the *immediate successor* of x. A *source* is a node without any predecessor, a *sink* is a node without any successor.

A directed graph $G' = (N', E')$ is called a *subgraph* of G if and only if $N' \subseteq N$ and $E' \subseteq E$. For $N' \subseteq N$ the *restriction* of G onto N' is the subgraph $(N', E \cap (N' \times N'))$: this is that graph G' which consists of the subset N' of nodes from G and all those edges from G which connect the nodes of N'; the restriction is denoted by $G \mid N'$.

For $n \in \mathbb{N}$ a *path of length n* is a finite sequence of connected edges: $[(x_i, y_i) \in E \mid: 1 \le i \le n]$ with $y_i = x_{i+1}$ for all $1 \le i < n$; such a path is denoted by $p = (x_1, \ldots, x_n, y_n)$. The node x_1 and y_n are called *start* respectively *end node* of the path. The set of all paths of a graph G is symbolised by P_G. A path $p = (x_1, \ldots, x_n)$ is said to be *cyclic* if and only if there are $1 \le i < k \le n$ such that $x_i = x_k$; a cyclic path contains a subpath with identical start and end node. A directed graph G is called a *directed acyclic graph (DAG)* if and only if P_G doesn't contain any cyclic path.

Let $G = (N, E)$ be a directed graph and extend E by its inverse relation E^{-1} to form an Graph $G' = (N, E \cup E^{-1})$: G is called *connected* if and only if for any pair of different nodes $x, y \in N$ there is a path $p \in P_{G'}$ which connects both of these nodes, that is $p = (x, \ldots, y)$.

A directed graph $G = (N, E)$ which contains exactly one sink s and where for each $x \neq s$ there is exactly one path $p \in P_G$ with $p = (x, \ldots, s)$ is called a *tree*. The sink may be called a *root*, and sources may be called *leafs*. Trees are acyclic and connected.

Let $G = (N, E)$ be a DAG: For each $x \in N$ the *backward closure* of x in G, denoted by $B_G(x)$, is defined as the set of all nodes $y \in N$ with $x = y$ or for which a path $p \in P_G$ with $p = (y, \ldots, x)$ exists. This is the set of all nodes from which x can be reached.

Let $[N_i \mid: 1 \le i \le n]$, $n \in \mathbb{N}$, be a partitioning of the node set of a DAG $G = (N, E)$ – thus $N_i \cap N_k = \varnothing$ for $i \neq k$ and $N = \bigcup [N_i \mid: 1 \le i \le n]$: a *partitioning* of G is the sequence $[G \mid N_i \mid: 1 \le i \le n]$ of subgraphs restricted onto the sets N_i, $1 \le i \le n$. Each edge $(x, y) \in E$ with $x \in N_i$, $y \in N_k$ and $i \neq k$ is called a *cut*. It is not possible to partition a connected DAG into more than one independent subgraphs without any cut.

3. WATER CATCHMENTS AS DIRECTED ACYCLIC GRAPHS

For the sake of modelling water catchments are subdivided or classified by various hydrological or topological criteria. This yields a patchwork of adjacent subareas on the one hand and a set of classes well-defined by the applied criteria on the other hand. Each class determines a certain class characteristic behaviour of the subareas belonging to it. However the subareas of a class differ not only by location but are distinguished too by several non-classifying criteria defined by parameters.

To cope with this heterogeneous arrangement of areas the calculation of subareas one by one might be more appropriated than the generalised modelling based on classes. This distributed modelling requires the definition of data exchange – that is groundwater flow, interflow, and surface flow – between the subareas according to their topological properties like elevation, exposition, and slope: on the patchwork of areas a network of routing channels is specified, which determines how the water runs through the catchment [STAUDENRAUSCH, 1997]. By this way one can examine source, amount, portion and chemical composition of the water flowing through the areas. The hydro-dynamical routing network fixes the order of computation by defining the sources from where each area requires its input that is to calculate before the

area itself can be computed.

Relating this computational model to graph theory as outlined above, subareas correspond to nodes while routing channels between two adjacent areas are represented by edges between the nodes in question where the source of a flow is the predecessor of its succeeding destination. A source node refers to an area where water flow is induced and a sink to an outlet of the water catchment. A sub-catchment is represented by the restriction of the whole graph onto the set of nodes corresponding to the areas belonging to it. Paths stand for water flow through a series of subareas. A backward closure of a node may be interpreted as the set of all areas from where the area referred by that node gets its input that is those areas it depends on.

Supposing a well-defined routing network, it is reasonable to assume that the corresponding graph – directed by definition – is not only acyclic but connected too. Regardless of some exception, it is also obvious that such graphs tend to match the definition of trees: compared to confluences, channel branches are rather seldom.

To partition the graph means to coarsen the original subdivision of the catchment into non-overlapping and not necessary coherent areas covering the whole catchment. Each of these areas is represented by an particular partition and consists of those subareas whose corresponding nodes are assigned to its partition. A cut represents water flow between such areas.

4. GRAPH PARTITIONING: THE PROBLEM AND ITS COMPLEXITY

A directed acyclic graph is well defined by a set N of nodes and a function $s: N \mapsto \mathfrak{P}(N)$ which assigns to each node the set of its immediate successors. By this function a dual function $p: N \mapsto \mathfrak{P}(N)$, $n \in s(m) \Leftrightarrow m \in p(n)$ for $m, n \in N$, returning the set of a node's predecessors is uniquely determined. It is assumed that N is a finite set.

Moreover one may want to apply an additional function $c_N: N \mapsto \mathbb{N}$ defining the internal complexity of the computation of a particular node: the higher the integer assigned to a node the higher the computational complexity of the corresponding area. The value $c_N(n)$ is said to be the *cost* of a node $n \in N$. The costs of a set of nodes will be calculated by summing up the costs of all nodes belonging to that set; by this way the costs function is extended: $c_N: \mathfrak{P}(N) \mapsto \mathbb{N}$.

The number of partitions is given by $k \in \mathbb{N}$. It is assumed that k is less than or equal to the number of N. If the graph will be partitioned for the sake of distributing its nodes among several heterogeneous processors k reflects the number of processors. In this case a further function $c_p: P \mapsto \mathbb{N}$ which defines the computational power of each processor may be used: the higher the c_p-value of a processor the higher its computational power.

In general it might be necessary to define a function which weights the edges between nodes. In the special case of water catchment simulations where edges reflect the water flow between subareas it is reasonable to avoid this additional complexity: the connection of two nodes just stands for the transfer of data; compared to the internal calculations a quite simple operation which is moreover homogeneously complex within the whole network of nodes.

From the given data the following can be computed: the set of paths from a node to a sink or to a source, the length of a particular path as well as the mean, maximal and minimal length of paths to a sink or to a source, the costs of a subgraph by summing up the costs of the nodes it contains, and the *total costs of calculation* of a node as the costs of its backward closure.

While partitioning the graph two constraints are to be matched:

> C1. The computational load of the partitions should be balanced by taking into account the costs of the nodes being assigned to a particular partition and the computational power of the partition's processor.
> C2. The number of cuts, that is the number of edges between different partitions, should be balanced too and as limited as possible to minimise the need for inter-process-communication.

Due to its combinatorial nature the problem of partitioning a graph as described above is known to be NP-complete [GAREY et al., 1976], that is, the number of the set of its systematic solutions is growing exponentially with the linear increase of the number of nodes. In the most simple case where all nodes have the same complexity and all processors the same computational power there are about 10^{473}, 10^{596}, 10^{691}, 10^{981} respectively 10^{1753} possibilities to partition 1,000 nodes into 3, 4, 5, 10 respectively 100 clusters while the same partitionings applied on 10,000 nodes yield 10^{4766}, 10^{6013}, 10^{6979}, 10^{9976} respectively 10^{19704} solutions. Trying to solve the problem systematically means not only to generate all of these solutions but to estimate and compare their quality too.

In the current model of computation it is impossible to solve such complex problems systematically. That's why imperfect algorithms are preferred which depend significantly on the topology of the graphs in questions and provide sufficient optimal solutions in a reasonable amount of time [ABBAS, 1995; HENDRICKSON et al., 1995a/b; HUANG et al., 1996; KERNIGHAN et al., 1970; V. LASZEWSKI, 1993]. If additional knowledge of the structure of a graph or a class of graphs is given it can be employed to formulate an appropriate and sufficient optimized, as well as fast, algorithm.

5. GRAPH THEORETICAL CHARACTERISTICS OF WATER CATCHMENTS

Water catchments are underlied by natural channel system which usually consists of a main stream flowing into the outlet and several secondary affluents. In general this system is not as much branched as the catchment is subdivided: usually there are more areas than channels due to the fact that one criterion for distinguishing subareas is often an existing watershed and that there may be areas located above natural water sources without any canalised stream flowing through them.

Although the topographic structure of the channel system is not necessarily a criterion for the classification respectively subdivision of a catchment, it functions as a basic frame work or a kind of skeleton of the corresponding routing graph because the graph topological structure reflects the relation of water flow between subareas.

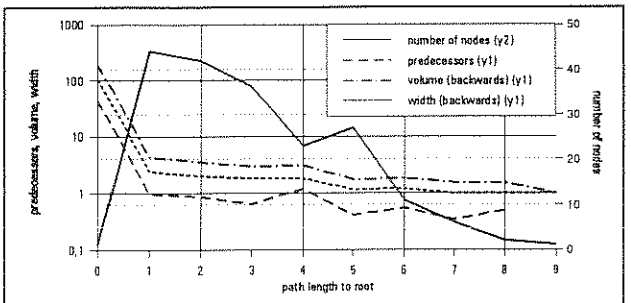

Figure 2: Properties of a catchment with 193 subareas

Depending on the regarded scale slope and exposition of those subareas through which the main stream and its tributaries flow don't change significantly. Thus each of these channels will be assigned to one or only few nodes which are connected to each other. Because the regional extent of such areas is quite large and lots of areas drain into them the corresponding nodes possess relatively many predecessors. All the other subareas are locally restricted and the amount of neighbours they have is limited: the relation of drainage from or to other areas is easy to survey and more linear than branching. That's why nodes near the outlet of the routing graph own a over-proportional amount of predecessors while remote nodes are arranged in more linear chains (fig. 1).

This theoretical consideration may be justified by examining a number of properties which can be measured in the corresponding graph: for a given node n assigned to an area s the *number of predecessors* stands for the number of areas draining into s, the *backward volume* for the number of the backward closure of n, that is the total number of areas draining into s, and the *backward width*

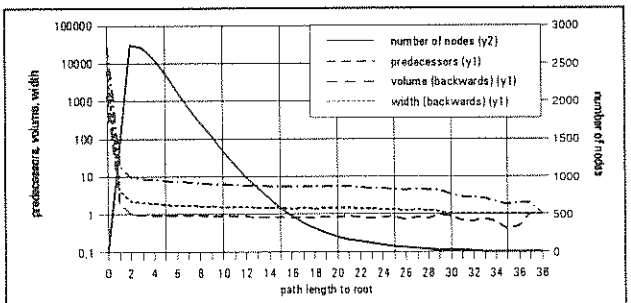

Figure 3: Properties of a catchment with 27121 subareas

for the number of sources from where water flowing through s is initially originated. As both of the figures 2 and 3 – illustrating the properties of the same catchment classified into 193 subareas by hydrological and into 27121 subareas by mere topographic criteria – show the mean of the number of predecessors, the backward volume and the backward width calculated over the set of nodes which have the same distance to the outlet decreases drastically behind the first three or four routing sections.

6. HEURISTICS AND ALGORITHMS

Especially the backward volume may be taken into account while developing an appropriate heuristic algorithm. For the sake of universality it may be replaced by the total cost of a node, supposing the complexity of calculations is evenly distributed among the graph. As explained above, routing graphs of water catchments are distinguished by usually consisting of many relatively slim and small subtrees all of them linked near the root of the whole graph.

Because the number of partitions is usually significantly lower than the number of the node set, the number of nodes assigned to a particular partition is rather high: thus each partition will consist of a bunch of those small subtrees which are linked to the root. The following algorithm copes with that conditions:

Precondition: let U be the set of all nodes which are not assigned to any partition, set it initially to the whole DAG $G = (N, E)$; let $p_1, \ldots, p_n$ be the set of initially empty but consecutively filled up partitions; normalise c_P by multiplying with $c_N(N) / \sum\{c_P(p_i) \mid: 1 \le i \le n\}$.
As long as $U \neq \varnothing$ continue:
1. Choose a partition p_i with greatest free capacity k_i where $k_i = c_P(p_i) - c_N(p_i)$.
2. Look for a node $n \in U$ with $|k_i - c_N(B_G(n) \cap U)| \le \varepsilon$, with a threshold $\varepsilon \ge 0$; if found go on with 4.
3. not found: choose a node n with $|k_i - c_N(B_G(n) \cap U)|$ minimal, distinguish 3.1 where $c_N(B_G(n) \cap U) < k_i$ or 3.2 where $k_i < c_N(B_G(n) \cap U)$ is allowed.
4. Add $B_G(n) \cap U$ to p_i, set U to $U \setminus B_G(n)$.

The normalisation of c_P is needed for the applied arithmetic operations to make sense: it garantees the overall equation $\sum\{c_P(p_i) \mid: 1 \le i \le n\} = c_N(N)$. The partition chosen in step 1 provides a free maximum capacity to ensure that the cluster of nodes to be assigned to it is as big as possible. The value $c_N(B_G(n) \cap U)$ stands for the total costs of a node regardless of those nodes which are already assigned to a partition: it reflects the computational costs still to be distributed. Step 2 looks up for a node which fits best into the free gap. If such a node doesn't exists step 3.1 chooses a node whose total costs is as high as possible but doesn't exceed the capacity of the partition; while step 3.1 allows capacity exceed. In step 4 all the nodes of the backward closure of the node chosen which are not already assigned are added to the partition in question and taken off the set of unassigned nodes.

The basic algorithm as it is sketched above has linear complexity: in the worst case the loop is to be performed for each node. The main disadvantages are the closure operation and the set intersection which can raise the complexity up to $a * \#N^b + c$ for fixed numbers a, b, and c: thus the overall time complexity of the algorithm is polynomial.

If the topology of the routing graph which is to be

partitioned is 'reasonable' in that sense that it is almost like a tree confirming to the above stated structure the version 3.1 garantees that the costs of the partitions are balanced according to the constraint C_1 while version 3.2 which allows that the partitioning costs may be exceeded is more suitable if constraint C_2 is preferred because it permits bigger node clusters (fig. 4).

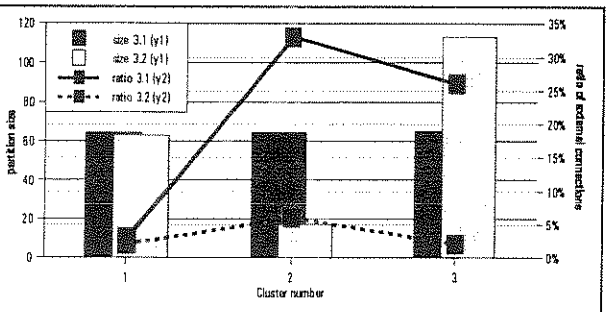

Figure 4: Partition sizes and percental portion of cuts depending on the algorithm alternatives 3.1 and 3.2 for the partitioning of 193 subareas into 3 partitions

7. OPTIMISATION AND EVALUATION

The algorithm as given above is a first approximation which provides a reasonable 'optimal' result if the regarded graph has a shape 'typical' for routing networks and the number of nodes is significantly higher than the number of partitions. Under these circumstances it is usually impossible to improve the result further on. Otherwise additional optimisations may be necessary: for example selective permutations of pairs of nodes belonging to different partitions, selective reassignments by path tracing to minimise the number of cuts on a certain path etc.

To evaluate a given partition the following criteria may be considered:

1. The constraints C_1 and C_2 should be satisfied.
2. The costs of inter-process-communication – the costs of cuts – should be minimal and balanced according to the given hardware.
3. The ratio of client to server connections, that are the cuts where a partition is on the successor respectively predecessor side, should be as minimal as possible per partition.

Depending on the weights one ascribes to these criteria the initial partitioning generated by the given algorithm and further refinements can be compared to each other to find the best solution suited for the regarded problem.

8. FURTHER REFERENCES

ABBAS, N.: Graph Clustering: Complexity, Sequential and Parallel Algorithms. University of Alberta: 1995. (Technical Report TR 95-01)

CRAWFORD, N. H.; LINSLEY, R. K.: *Digital Simulation in Hydrology: Stanford Watershed Model IV.* Dep. of Civil Engineering, Stanford University: 1966. (Technical Report 39)

GAREY, M.; JOHNSON, D.; STOCKMEYER, L.: Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science 1.* 237–267. 1976.

HENDRICKSON, B.; LELAND, R.: A Multilevel Algorithm for Partitioning Graphs. *Proceedings of Supercomputing '95.* San Diego: 1995a.

HENDRICKSON, B.; LELAND, R.: An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations. *SIAM Journal on Scientific Computing 16.* 452–469. 1995b.

HUANG, Y.-W.; JING, N.; RUNDENSTEINER, E. A.: Effective Graph Clustering for Path Queries in Digital Map Databases. *ACM 5th International Conference on Information and Knowledge Management CIKM '95.* 1996.

KERNIGHAN, B.; LIN, S.: An Efficient Heuristic Procedure for Partitioning Graphs. *Bell Systems Technical Journal 49.* 291–307. 1970.

V. LASZEWSKI, G.: A Collection of Graph Partitioning Algorithms: Simulated Annealing, Simulated Tempering, Kernighan Lin, Two Optional, Graph Reduction, Bisection. *NPAC Technical Report SCCS-477.* 1993.

LAU, L.: *Implementation of Scientific Applications on Heterogeneous Parallel Architectures.* University of Queensland, Brisbane, Australia: 1996.

SINGH, V. P.: Watershed Modeling. *Computer Models of Watershed Hydrology.* 1-22. 1995. (Water Resources Publications)

STAUDENRAUSCH, H.: Entwicklung und Evaluierung von Netzwerktopologien für homogene hydrologische Modelleinheiten durch GIS-Methoden. *Angewandte Geographische Informationsverarbeitung IX.* 405–410. Salzburg: 1997. (Salzburger Geographische Materialien 26)