On exact algorithms for mapping communicating tasks onto heterogeneous systems

Bora Uçar

CERFACS, Toulouse, France

CCGSC08, 14–17 Sep 2008, Flat Rock

Much work in progress (jointly with Kamer Kaya, Bilkent Univ., Ankara, Turkey)

Financé par ANR

Supported by Agence Nationale de la Recherche through SOLSTICE project number ANR-06-CIS6-010.

Outline

The assignment problem

Computing model: Heterogeneous processors and network.

Application model: Communicating tasks modeled using a task interaction graph, in short TIG (vertices represent tasks and edges represent intertask communications). No precedence relation among the tasks.

Objective function: Minimize the sum of the total execution and communication costs (optimize system utilization).

Formulation

Notation

$\mathcal{P}$: The set of P processors,

$\mathcal{T}$: The set of T tasks,

$\{x_{tp}\}_{T \times P}$: Expected time to compute matrix (ETC); x_{tp} denotes the execution cost of task t on processor p,

$G = (\mathcal{T}, E)$: Task interaction graph; edge $(t, u) \in E$ is associated with a communication cost multiplier c_{tu} which incurs when the tasks t and u are assigned to different processors.

$\{d_{pq}\}_{P \times P}$: The distance between processors, i.e., if the tasks t and u are assigned to processors p and q, then a communication cost of $c_{tu} \times d_{pq}$ is incurred. Symmetric, i.e., $d_{pq} = d_{qp}$; and $d_{pp} = 0$.

Formulation (cont')

Optimization problem

Find an assignment $\mathcal{A} : \mathcal{T} \rightarrow \mathcal{P}$ that minimizes the sum of execution and communication costs:

$$\min \left(\sum_{t=1}^{T} \sum_{p=1}^{P} a_{tp} x_{tp} + \frac{1}{2} \sum_{(t,u) \in E} \sum_{p=1}^{P} \sum_{q=1}^{P} a_{tp} a_{uq} c_{tu} d_{pq} \right) \textit{ subject to}$$

$$\sum_{p=1}^{P} a_{tp} = 1, \quad t \in \mathcal{T}$$

$$a_{tp} \in \{0,1\}, \quad p \in \mathcal{P}, \quad t \in \mathcal{T}$$

Here, if task t is assigned to processor p, then $a_{tp} = 1$ and 0 otherwise.

Background

The general problem is NP-complete [references in Bokhari, IEEE TSE (1981)].

Polynomial-time solvable instances

Two processor systems [Stone, IEEE TSE (1977)], in the time complexity of maximum-flow algorithm,

TIGs in tree structure [Bokhari, IEEE TSE (1981)] on heterogeneous networks, in $O(TP^2)$ time; on homogeneous networks [Billionnet, IEEE TPDS (1994)], in $O(TP)$ time,

TIGs in series-parallel graph structure [Towsley, IEEE TSE (1986)], in $O(TP^3)$ time,

TIGs in partial k-tree structure [Fernandez-Baca, IEEE TSE (1989)], in $O(TP^{k+1})$ time.

Some recent works

Heuristics

- Minimize the completion time [Arafeh, Day, and Touzene, JSA (2007)],
- Minimize the total cost [U., Aykanat, Kaya, and Ikinci, JPDC (2006)],
- Total communication cost in heterogeneous network [Orduña, Silla, and Duato, JSA (2004)],
- Homogeneous processors and heterogeneous network [Senar, Ripoll, Cortés, and Luque, JSA (2003)],

Exact algorithms

- Homogeneous processors, heterogeneous network [Ma, Chen, and Chung, JPDC (2004)],
- Heterogeneous processors and network [Tom and Murty, Sys. Soft. (1999)],
- Heterogeneous processors, homogeneous network [Kafil and Ahmad, IEEE Concurrency (1998)].

Our aim and contributions

Aim

Exact algorithms for small instances (problem size is small but the search-space is huge, P^T),

- performance is of utmost importance,
- can be used to evaluate heuristic algorithms.

Contributions

- An exact algorithm using A^*-search,
- Use of graph theoretical concepts to reduce the search-space size,
- Use of polynomial time exact algorithms within the A^*-search.

A^*-search (General)

A^* is a best-first, graph search algorithm [Russell and Norvig, AIMA (2003)].

It finds a least cost path from a given initial node to a goal node.

Evaluation function of a node v

$$f(v) = g(v) + h(v)$$

- $g(v)$ is the actual cost to reach the node v from the initial node,
- $h(v)$ is the estimated cost to a goal node from v.

The function $h(v)$ should be an *admissible* heuristic, i.e., should never overestimate the actual cost from v to a goal node.

A node with the minimum f value is expanded, i.e., all of its successors are generated and the f value for each one is computed.

A^*-search (Task assignment)

Search-space is in the form of a tree

Initial node: no assignment; Goal nodes: all tasks are assigned; Intermediate node $v = \langle t, p \rangle$ at level t: the decision of assigning task t to processor p is appended to the partial solution.

Initialization: P nodes

- $\langle 1, p \rangle$ for $1 \leq p \leq P$, corresponding to the assignment of task 1 to the processor p.
- $g(1, p) = x_{1p}$ and $h(1, p)$ is an estimate of the cost of assigning the remaining tasks to the processors with the information that the task 1 is assigned to the processor p.

Expanding $\langle t, q \rangle$—has minimum f value

P nodes $\langle t + 1, p \rangle$, for $1 \leq p \leq P$ are created, g and h values are computed and the nodes are inserted into a list using f as a key.

The functions $g(t+1,p)$ and $h(t+1,p)$

The actual cost, $g(t+1,p)$, to a node

Easy to formulate using the parent node $\langle t, q \rangle$:

$$g(t+1,p) = g(t,q) + x_{t+1,p} + comm([1,\ldots,t], t+1)$$

Admissible heuristic $h(t+1,p)$

Proposal: use Bokhari's exact algorithm for the TIGs in tree structure.

Suppose we want to compute $h(t+1,p)$ as a lower bound for the cost of assigning tasks $t+1$ to T (the tasks 1 to t are assigned as in node $\langle t, p \rangle$).

Consider any spanning tree/forest of the tasks with id $t+1$ to T. Run Bokhari's algorithm on this structure; the cost is a lower bound on the cost (some adjustments are necessary).

Constructing the tree for $\langle t, p \rangle$

Edges (s, u) for $s < t \leq u$

The task s has been assigned, say to p. Perform a set of updates:

$$x'_{uq} = x_{uq} + c_{su} d_{pq} \quad \text{for each } q$$

such that the updated costs account for the communications involving already assigned tasks (delete/nullify those edges).

Other edges (u, v) for $t \leq u, v$ kept intact; no change is necessary.

For each node of the search-space, build a tree structured problem.

Constructing the trees for Bokhari's algorithm

Bookkeeping

Build a spanning tree/forest $S(t)$ of the vertices from t to T, for each t at the beginning of the algorithm—maximum weighted spanning tree.

During the expand operation

For each node $\langle t+1, p\rangle$ of the search-space get the tree $S(t+1)$, do the necessary transformations and solve it optimally.

This is an admissible heuristics: the assignment found by the tree algorithm is the best for the remaining tasks and the subset of communications taken into account.

Preprocessing: Ordering the tasks (1)

We can stop when a node with an exact h has the minimum key f.

If at a search-space node $\langle t, p \rangle$ we know that the remaining tasks forms a tree/forest, h is exact.

Order the tasks

Order a set of tasks with acyclic inter-task connectivity as the last tasks.

If F is such a set, we reduce the search-space size from P^T to $P^{(T-F)}$.

Finding the maximum cardinality acyclic subgraph in a graph is an NP-hard problem (minimum feedback vertex set). For now, we use heuristics without any guarantee [Bafna, Berman, and Fujito, SIAM JDM (1999) has an algorithm with approximation guarantee.]

Preprocessing: Ordering the tasks (2)

We have found the last F tasks.

Order of the other tasks

According to their h value in increasing order.

Rationale: During the search, g tends to increase and h becomes more accurate as we go deeper in the search-space.

We tried some others as well (including MaxMin), but the above one looks better.

Summary

A^* for task assignment

1: Find a maximal acyclic set F of vertices
2: Order the other vertices
3: Initialize a priority queue Q with P nodes $\langle 1, 1\rangle, \langle 1, 2\rangle, \ldots, \langle 1, P\rangle$ with key $f = g + h$
4: **while** $Q \neq \emptyset$ **and** t of $\mathrm{first}(Q) \notin F$ **do**
5: $\quad \langle t, p\rangle \leftarrow \mathrm{extractMin}(Q)$
6: $\quad$ Create P nodes $\langle t+1, 1\rangle, \ldots, \langle t+1, P\rangle$, computing f, g, and h; insert into Q
7: **end while**
8: Complete the solution with the h of $\mathrm{first}(Q)$ $\triangleright$ as $\mathrm{first}(Q)$ contains a $t \in F$

- no more than 50Million nodes in Q,
- constructive heuristic to get an upper bound U.

A^* never expands a node with $f > C^\star$ where $C^\star$ is the optimal assignment cost.

Not inserting the nodes with $f > U$ can reduce memory requirements (does not help to reduce time though).

Experiments

Set up

- TIGs: small sparse matrices (lacking real-life applications) $T = \{59, 72, 87, 209, 307\}$ tasks.
- Communication: random integers from 1–100.
- ETCs are obtained using standard methods [Ali, Siegel, Maheswaran, Hensgen, and Ali, HCW2000],
    - ETC0: low task, low machine heterogeneity,
    - ETC1: low task, high machine heterogeneity,
    - ETC2: high task, low machine heterogeneity,
    - ETC3: high task, high machine heterogeneity.

  Scaled: communication-to-computation ratio $\rho = \{0.7, 1.0, 1.4\}$.
- $P = \{2, 3, 4, 8\}$ processors.

Created 3 random instances for each T, P, ρ triplet.

Experiments

An alternative heuristic h from the literature

Ignore all communications, and assign the tasks to their best processor [Kafil and Ahmad, IEEE Concurrency (1998)].

As in the proposed heuristic function h, ordering a set of independent tasks last helps reduce the search-space (our add on)—h becomes exact.

Maximum independent set problem is NP-hard too.

Experiments

	h=Independent set				h=Tree			
	P				P			
T	2	3	4	8	2	3	4	8
59	$\surd$	$\surd$	$\surd$	E012-	$\surd$	$\surd$	$\surd$	$\surd$
72	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
87	$\surd$	E012-	E012-	E012-	$\surd$	$\surd$	$\surd$	E012-
209	E012-	E012-	E012-	E-1- -	$\surd$	E012-	E012-	E012-
307	E012-	E-1- -	E****	E****	E012-	E012-	E012-	E****

$\surd$: all instances are solved (all types of ETC and ρ, 3 random instances).

$E012-$: all instances with ETC type 0, 1, and 2 are solved. For ETC of type 3 some or all instances needed more than 50million nodes, hence exited without solving (for some ρ).

*: were still running when I left the office (may or may not obtain solution).

Experiments: Some comparisons with the independent set heuristic

Problem	Metric	h=Independent set	h=Tree
$T = 307 \quad P = 3$	Search-space size	3^{307-79}	$3^{307-147}$
ETC2 $\quad \rho = 1.4$	Opened nodes	22,917,457	22,190
	Time (s.)	4,858	4
$T = 59 \quad P = 8$	Search-space size	8^{59-22}	8^{59-38}
ETC3 $\quad \rho = 1.0$	Opened nodes	46,246,756	1849
	Time (s.)	10,617	0.4

(Runs are on a machine with 64 AMD Dual 250 Opteron.)

Reminders

- ETC2: high task, low machine heterogeneity.
- ETC3: high task and machine heterogeneity.
- ρ: communication-to-computation ratio.

Conclusion and future work

Aiming at solving problems of size $T \geq 500$ and $P \geq 4$ (hopefully on real life applications).

Plans

- implement memory efficient variants of A^*,
- use constructive heuristics to find upper bounds at different levels to save some memory (like branch-and-bound algorithm),
- implement the exact algorithms for other special cases of the problem (e.g., series-parallel graphs) to have a better heuristic function h.
- the quadratic assignment problem bears similarities. Investigate the applicability there (the largest instance in QAPLIB has a size of $n = 250$, most are smaller than 100, not all have been solved optimally—this time search-space is usually of size n^n).

Further information

Thank you for your attention.

`http://www.cerfacs.fr/algor`

`http://www.cerfacs.fr/~ubora`
`ubora@cerfacs.fr`