Nesting OpenMP in MPI to Implement a Hybrid Communication Method of Parallel Simulated Annealing on a Cluster of SMP Nodes

Agnieszka Debudaj-Grabysz[1] and Rolf Rabenseifner[2]

[1]*Silesia University of Technology, Gliwice, Poland;*
[2]*High-Performance Computing Center (HLRS), Stuttgart, Germany*

OUTLINE

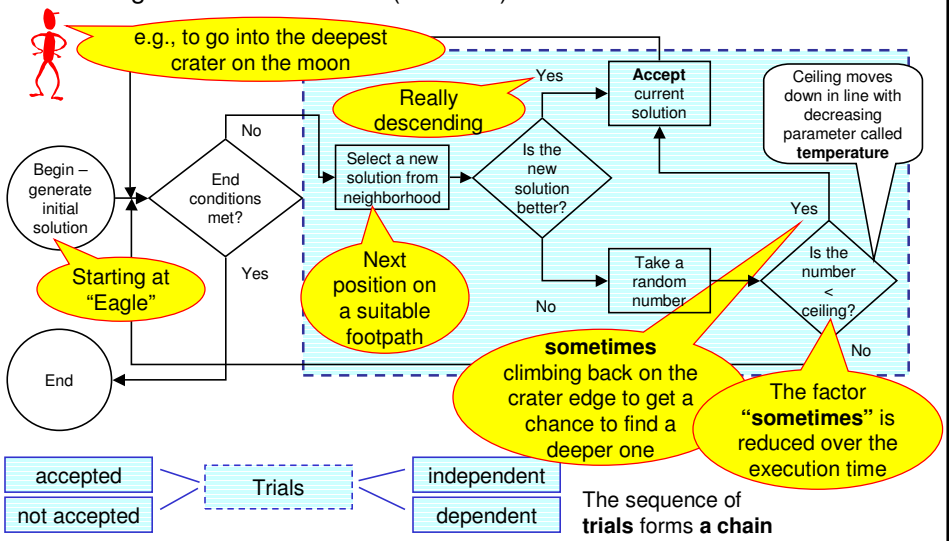
1. The algorithm of simulated annealing (SA)
2. Goals
3. Different approaches to MPI-parallel SA:
   - Intensive communication algorithm
   - Independent runs
   - Periodically interacting searches
4. Basic hybrid communication (HC) method – MPI & OpenMP
5. HC method with data exchange – an additional way to improve quality of results

Experimental results
→ advantages of HC methods over the other tested ones

THE GOAL AND ALGORITHM OF SIMULATED ANNEALING

Trial

Finding the state of minimal (maximal) value of the cost function



accepted
not accepted
Trials
independent
dependent

The sequence of **trials** forms **a chain**

Goals

- Using **more than 100 CPUs**.
- Keeping accumulated CPU time constant, i.e., **optimal parallel efficiency**.
- Modifying the parallel algorithm to **keep the best possible quality.**

Vehicle Routing Problem With Time Windows (VRPTW)

- *as an underling optimization problem*
- *to illustrate the features of tested methods*

MPI-PARALLEL SIMULATED ANNEALING

DECOMPOSITION:

The creation of random solutions (generating trials) is decomposed among processors.

The total number of generated trials is fixed.

COMMUNICATION:

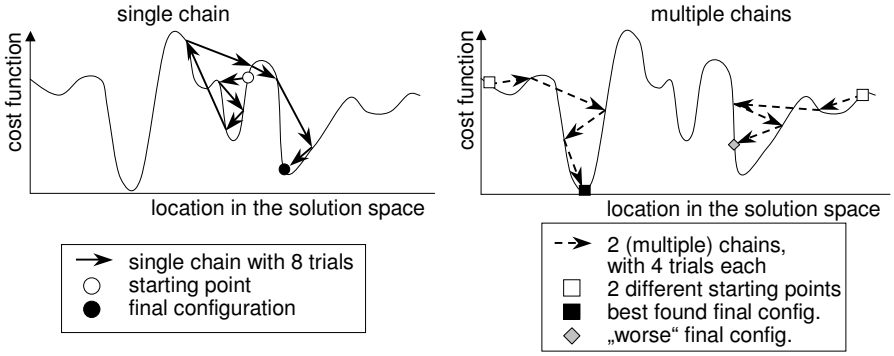
Possible requirement of broadcasting when an acceptable solution is found

MPI-PARALLEL SIMULATED ANNEALING

Possible intensity of communication in parallel SA

Communicating EACH event

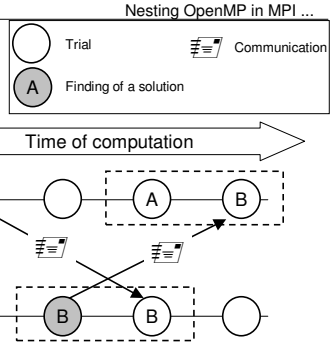
Communicating NO event

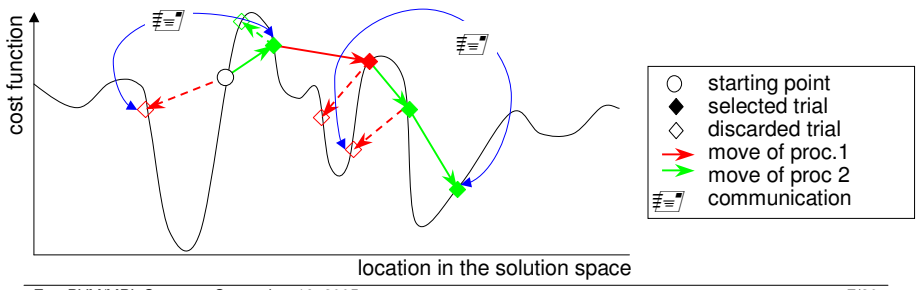


MPI-PARALLEL SIMULATED ANNEALING

Intensive communication algorithm

- Processor which finds a solution communicates it together with a **real time stamp** to the remaining processors **without synchronization**
- Processor which has to choose among a few solutions **picks the newest** according to the real time stamps

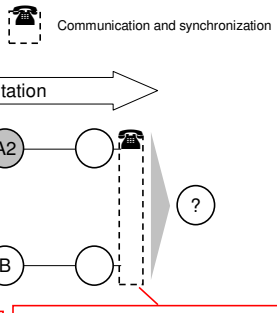


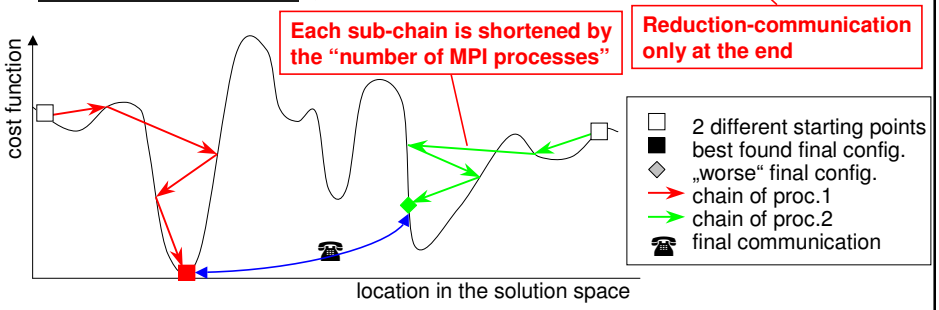


PARALLEL SIMULATED ANNEALING

Non-communication algorithm

- All available processors run basically sequential algorithms
- The original chain is split into subchains
- At the end the best solution is picked up as the final one

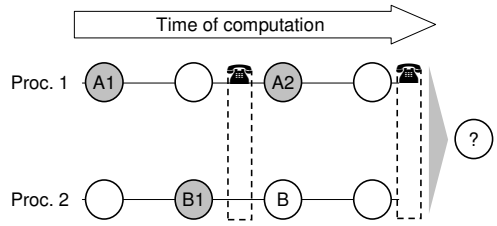


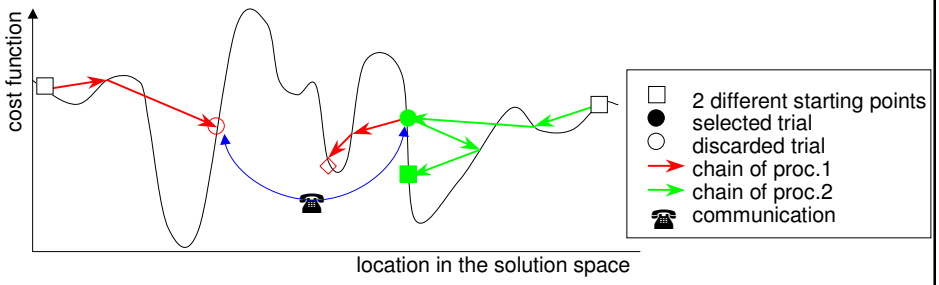


PARALLEL SIMULATED ANNEALING

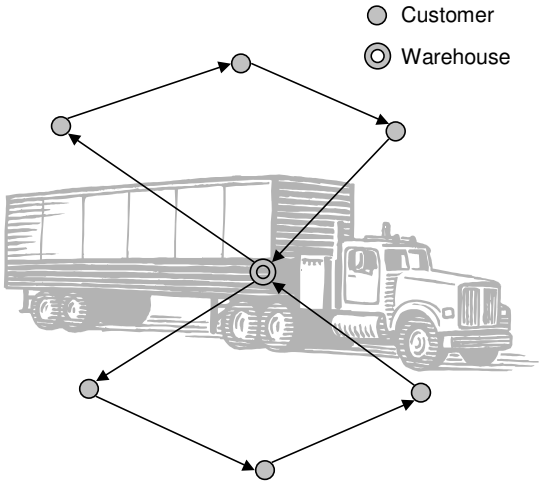
Lightweight communication – periodically interacting searches

- Processes communicate after performing a subchain called a *segment*
- The best solution is selected and mandated for all of them





VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

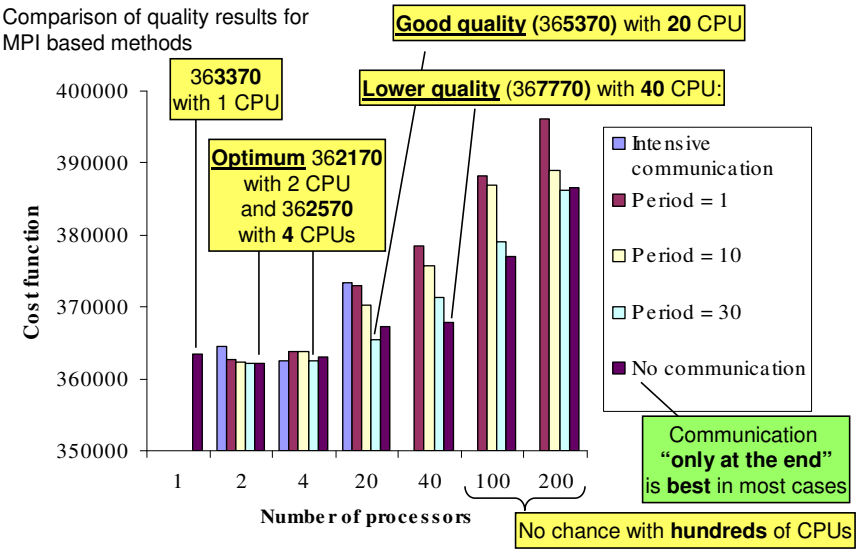


Goal
- Minimizing number of route legs
- Minimizing of the total travel distance

Constraints
- Customer and warehouse time windows
- Vehicle capacity
- Duration of servicing a single customer
- Customer's demand

EXPERIMENTAL RESULTS*

Comparison of quality results for MPI based methods

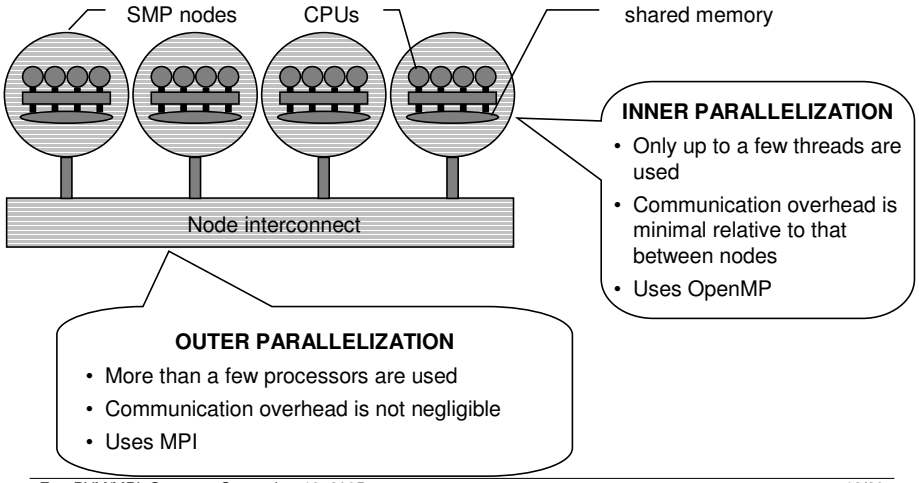


* Experiments carried out on NEC Xeon EM64T Cluster

Data file: R108

THE BASIC HYBRID COMMUNICATION METHOD

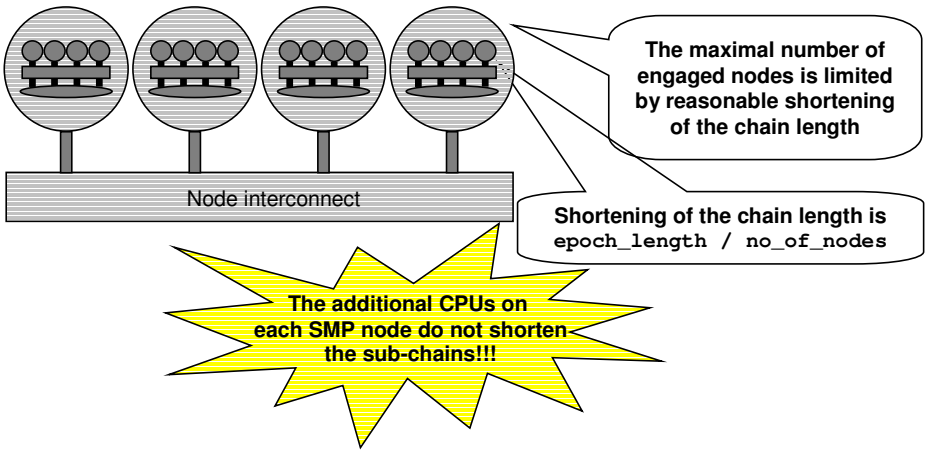
Hybrid character of the hardware → Parallelization uses two levels



THE BASIC HYBRID COMMUNICATION METHOD

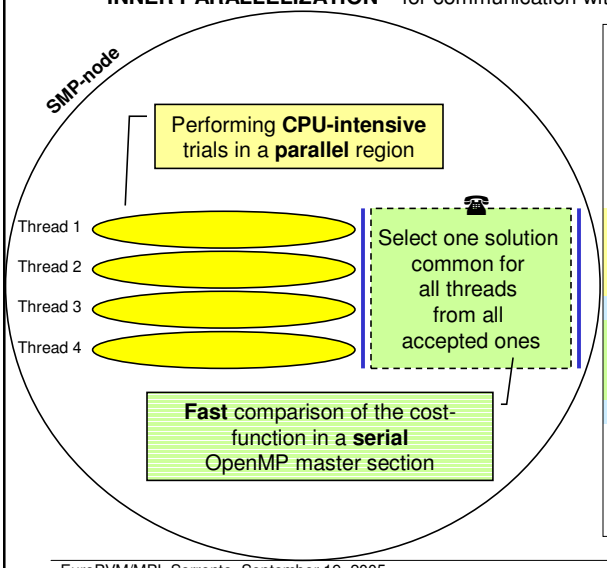
Using "MPI-communication only at the end" algorithm

OUTER PARALLELIZATION – for communication between nodes



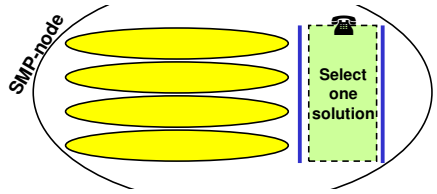
THE BASIC HYBRID COMMUNICATION METHOD

INNER PARALLELIZATION – for communication within nodes




```
for(l=0;l<NoOfTempRed;l++)
{
#pragma omp parallel private(ii)
{
 for(ii=0;ii<EpochLengthPerNode;
 ii=ii+SetOfTrialSize)
 {
#pragma omp for schedule(static)
  for(i=0;i<SetOfTrialSize;i++)
   perform_trial();
#pragma omp barrier
#pragma omp master
  {select one solution common
    for all threads ... }
#pragme omp barrier
 } /*end for (ii=0;...) */
} /*end of parallel region*/
 {reduce the temperature ... }
} /*end for(l=0;...)*/
```

THE BASIC HYBRID COMMUNICATION METHOD




```
for(l=0;l<NoOfTempRed;l++)
{
#pragma omp parallel private(ii)
{
 for(ii=0;ii<EpochLengthPerNode;
 ii=ii+SetOfTrialSize)
 {
#pragma omp for schedule(static)
  for(i=0;i<SetOfTrialSize;i++)
   perform_trial();
#pragma omp barrier
#pragma omp master
  {select one solution common
     for all threads ... }
#pragme omp barrier
 } /*end for (ii=0;...) */
} /*end of parallel region*/
 {reduce the temperature ... }
} /*end for(l=0;...)*/
```

Optimization of the OpenMP parallelization:

Optimal "`SetOfTrialSize`":

1. As **large** as possible **to minimize load imbalance**.
2. As **short** as possible **to keep good properties** of the simulated annealing algorithm.

For effective OMP parallelization

1. the parallel threads must be forked and joined **only once** per temperature-epoch
2. each thread has to use its **own, independent random number generator.**

With `SetOfTrialSize` = 20 *(with VRPTW, R108)*

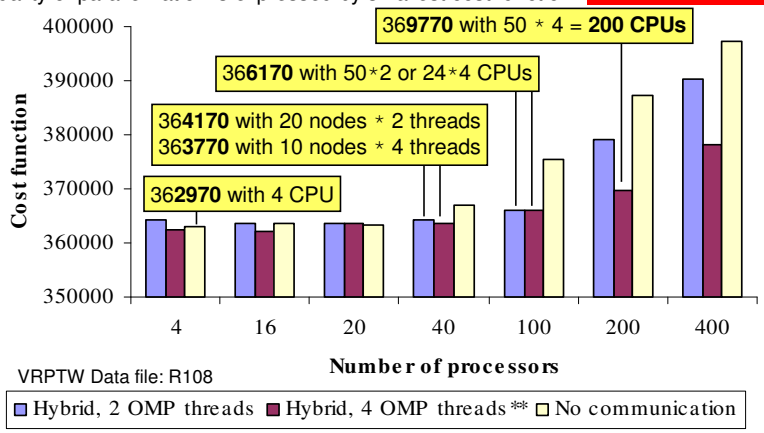
→ parallel OpenMP **efficiency** ~ 85% with 2 threads, and ~ **67%** with **4** threads.

EXPERIMENTAL RESULTS*

Comparison of quality results for **hybrid MPI+OpenMP** and independent runs methods:

- **The accumulated CPU time is kept constant.**
- Quality of parallelization is expressed by smallest cost function

And now we are 200x faster than with 1 CPU



VRPTW Data file: R108

* Experiments carried out on NEC Xeon EM64T Cluster

** Emulated usage of 4 OMP threads, based on results of tests of OMP parallelization carried out on NEC TX-7 system

HYBRID COMMUNICATION METHOD WITH DATA EXCHANGE

NEW RESEARCH

The idea

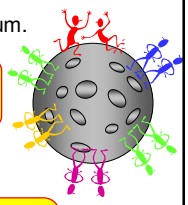
Incorporating one data exchange after elapsing a percent of the specified time limit (e.g. 50%, 70%)

During the exchange of the data the best solution is selected and mandated for all processes

The idea gives the possibility of:

- Heavy exploration of the search space during the first phase, i.e., a few (but only a few) paths can reach the area of the global minimum.

Many small groups of astronauts are looking independently for the deepest crater and hopefully, at least one group (=SMP node) is finding it

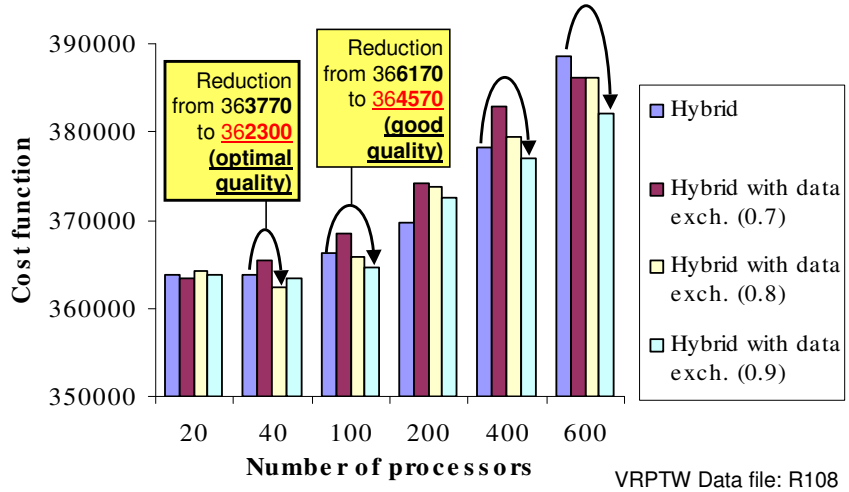


- Improvement of the best path during the second phase by all working processes (instead of only a few)

A short time before returning to earth, all groups are concentrated to the deepest crater found up to now.
The last minutes, they **all** try to find the deepest location in that crater!

EXPERIMENTAL RESULTS with 4 threads

Comparison of quality results for basic hybrid and hybrid with data exchange methods



VRPTW Data file: R108

Acknowledgements

- This project was supported by HPC-Europa.
- The research stay at HLRS Jan. – Feb. 2005 was granted by HPC-Europa.

HPC-Europa www.hpc-europa.org

Pan-European Research Infrastructure on High Performance Computing at

– HLRS, Stuttgart – SARA, Amsterdam – idris, Paris
– epcc, Edinburgh – CEPBA, Barcelona – CINECA, Bologna

Research Grants for Computational Science

- Access to HPC
- Scientific Collaboration
- Technical Support
- All **travel and living expense fully reimbursed!**
- 3 – 13 weeks

Poster outside

Applications are welcome at any level, from postgraduate researchers to senior profs.

- **Acceptance rate currently ~ 70 %**

Summary

- **Parallelization with periodic MPI communication**:
  - **Optimal quality** of the result (cost function cf< 36**3000**) with up to **4** CPUs
  - **Good** quality (cf< 36**5500**) with up to **20** CPUs
  - **Lower** quality (cf<36**9999**) with up to **40** CPUs

With **hybrid MPI+OpenMP** parallelization :

- **Good** quality (cf< 36**5000**) with up to **40** CPUs
- **Medium** quality (cf< 36**7000**) with up to **100** CPUs
- **Lower** quality (cf<36**9999**)
  - up to **200** CPUs (with **4 threads** per SMP node)

- **Hybrid MPI+OpenMP** (with 4 threads per SMP node) with an **additional (time based) communication step** after 80% or 90% of the available execution time:
  - **Good** quality (cf< 36**5000**) with up to **100** CPUs
  - **Optimal** quality (cf< 36**3000**) with up to **40** CPUs