Comparing Solution Methods for the Machine Reassignment Problem

Deepak Mehta Barry O'Sullivan Helmut Simonis

Cork Constraint Computation Centre
University College Cork, Ireland

October 2, 2012

Motivation

- Date centres are used in every segment of human activity such as telecommunications, internet, banks, entertainment, urban traffic.
- Western Europe data centres consumed 56 Tera-Watt Hours (in 2007) (TWh) of power, which is expected to almost double by 2020.
- A typical optimisation challenge is to keep machines well utilised such that the (power) costs are minimised.
- The 2012 ROADEF/EURO challenge in collaboration with Google is dedicated to machine reassignment problem, which is a common task in virtualisation and service configuration on data centres.

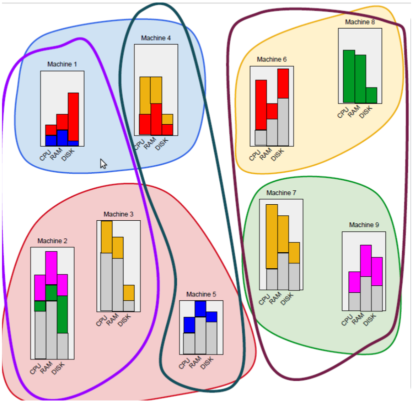
Outline

- Problem Description
- Solution Method
- Competition Results
- Conclusions

Machine Reassignment Problem

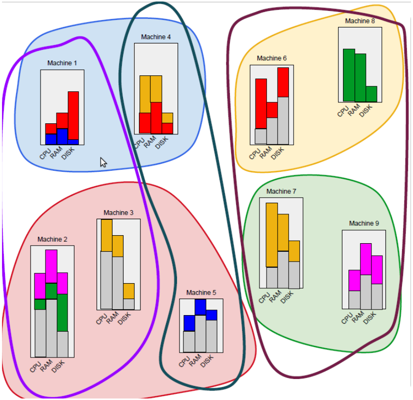
The machine reassignment problem is defined by a set of machines and a set of processes

- Each machine is associated with a set of (transient) resources, e.g. CPU, RAM, DISK
- Each process is associated with a set of required resource values and a currently assigned machine



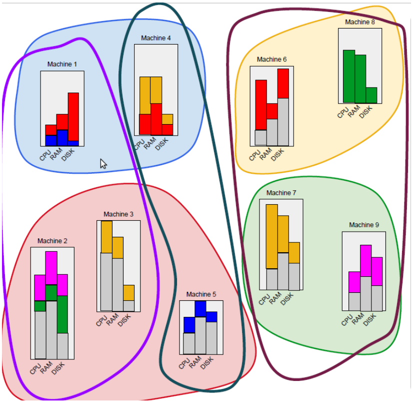
Machine Reassignment Problem

The objective is to improve the utilisation of the machines, as defined by a cost function, by reassigning the processes to machines while respecting a set of constraints



Machine Reassignment Problem: Constraints

- Capacity: The usage of a resource by a machine should not exceed its capacity.
- Conflict: A service is a set of processes. The processes of a service should be assigned to different machines.
- Spread: A location is a set of machines. The processes of a service should be spread over at least a given number of locations.
- Other: Dependency constraints, Transient Usage constaints



Machine Reassignment Problem: Costs

The objective is to minimize weighted sum of a load cost, a balance cost and several move costs.

- Load Cost: Any use of a resource by a machine above a given safety limit incurs a cost.
- Balance Cost: To balance the availability of resources.
- Process Move Cost: The cost of moving a process from a machine to any other machine.
- Service Move Cost: To balance the movement of processes among services.
- Machine Move Cost: The cost of moving any process from one machine to another machine.

Machine Reassignment Problem: ROADEF

Challenge Specific

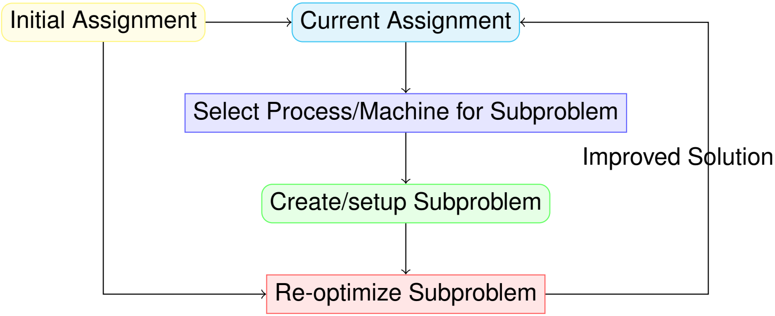
No. of Machines	**5000**
No. of Processes	**50000**
No. of Resources	**20**
No. of Services	5000
No. of Locations	1000
No. of Neighborhoods	1000
No. of Dependencies	5000
Time limit	**300 seconds**
space limit	4GB RAM

Problem Formulation

- Mixed Integer Programming (MIP)
  - solvers: GUROBI, CPLEX, etc.
  - space requirement exceeds 4GB for very large size instances
- Constraint Programming (CP)
  - solvers: CHOCO, GECODE, ECLIPSE etc.
  - problem can be formulated more concisely
  - The memory requirement never exceeded 300 MB

Solution Method: Large Neighborhood Search

Large Neighborhood Search (LNS) combines the power of *Systematic Search* with scaling of *Local search*



Principles of the LNS approach

LNS: Select Process/Machines for Subproblem

Observation

Selecting a set of processes from only some machines for reassignment works better than selecting only a few processes from many machines.

Two steps for selecting processes for reassignment:

1. select a subset of machines M'
2. select a subset of processes P' such that they are currently assigned to machines in M'

Two variants:

1. **Closed subproblem:** $\forall p \in P'$ the domain of machines is set to M'.
2. **Open subproblem:** $\forall p \in P'$ the domain of machines is set to M.

LNS: Create/setup subproblem

Create full problem model in memory

- At each iteration reinitialize the domains,
- Reassign the current machines to the processes which are not chosen for reassignment, and
- Perform constraint propagation

LNS: Create/setup subproblem

Create full problem model in memory

- At each iteration reinitialize the domains,
- Reassign the current machines to the processes which are not chosen for reassignment, and
- Perform constraint propagation

Create only a subproblem model in memory

- At each iteration create a subproblem in memory by projecting the problem on the processes selected for reassignment, and
- Perform constraint propagation

LNS: Create/setup subproblem

Create full problem model in memory

- At each iteration reinitialize the domains,
- Reassign the current machines to the processes which are not chosen for reassignment, and
- Perform constraint propagation

Create only a subproblem model in memory

- At each iteration create a subproblem in memory by projecting the problem on the processes selected for reassignment, and
- Perform constraint propagation

Setting up the domains could be time consuming

- The size of the problem instance is very large
- The time for solving is very limited, and
- The size of the subproblem is considerably smaller than the size of the input problem (e.g. $|P| = 50000$ and $|P'| = 100$)

LNS: Create/setup subproblem

Create full problem model in memory

- Unassign current machines from the selected processes
- Replenish the domains via incremental recomputation

LNS: Create/setup subproblem

Create full problem model in memory

- Unassign current machines from the selected processes
- Replenish the domains via incremental recomputation

The existing CP solvers do not provide this kind of support for LNS

LNS: Create/setup subproblem

Create full problem model in memory

- Unassign current machines from the selected processes
- Replenish the domains via incremental recomputation

The existing CP solvers do not provide this kind of support for LNS

Example

- Let $|P| = 50000$ and $|P'| = 100$
- Reinitialize all the domains, assign the current machines to 49900 processes, and perform constraint propagation,
- Unassign 100 processes and carefully add a set of removed values in the current domains by testing their validity with constraints

MIP-LNS

- A set of machines, M', is selected by solving a small optimization problem
  - the objective is to maximise the difference between
  - the current costs of the selected machines and
  - the costs resulting from the best possible utilization of the machines
- Subproblem is created in memory at each iteration
- CPLEX is used for re-optimizing a subproblem with 10 seconds time-out

CP-LNS

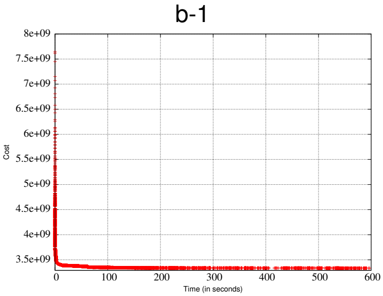
- M' is selected randomly, $1 \leq |M'| \leq 10$ and $|P'| \leq 40$.
- Full problem model is maintained, and at each iteration the domains are replenished incrementally via recomputation.
- Branch and Bound search with threshold on the number of failures
- Variable ordering heuristic for selecting a process
  - maximum increment in the objective cost when assigned a best machine
  - maximum total weighted requirement of a process
  - minimum number of machines available
- Value ordering heuristic for selecting a machine
  - minimum increment in the objective cost
- All the algorithms are implemented in C
  `http://sourceforge.net/projects/machinereassign/`

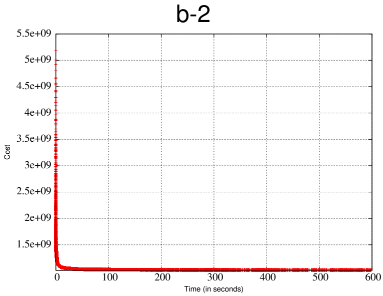
Experimental Results: Set A

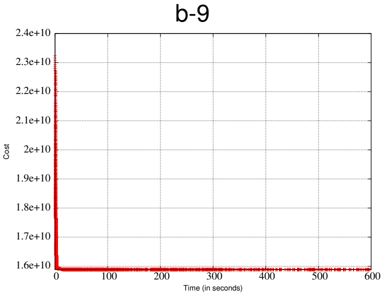
Results for set A obtained within 300 seconds

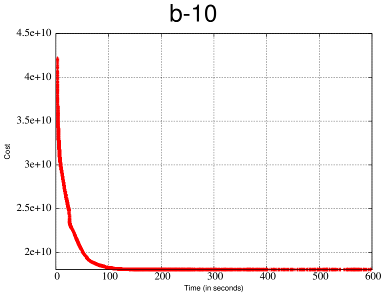
Prob	Initial	Best ROADEF	MIP-LNS	CP-LNS
a1-1	49,528,750	44,306,501	**44,306,501**	**44,306,501**
a1-2	1,061,649,570	777,532,896	792,813,766	**778,654,204**
a1-3	583,662,270	583,005,717	583,006,527	**583,005,829**
a1-4	632,499,600	252,728,589	258,135,474	**251,189,168**
a1-5	782,189,690	727,578,309	**727,578,310**	727,578,311
a2-1	391,189,190	198	273	**196**
a2-2	1,876,768,120	816,523,983	836,063,347	**803,092,387**
a2-3	2,272,487,840	1,306,868,761	1,393,648,719	**1,302,235,463**
a2-4	3,223,516,130	1,681,353,943	1,725,846,815	**1,683,530,845**
a2-5	787,355,300	336,170,182	359,546,818	**331,901,091**

Experimental Results: Set B









Competition Results: Teams

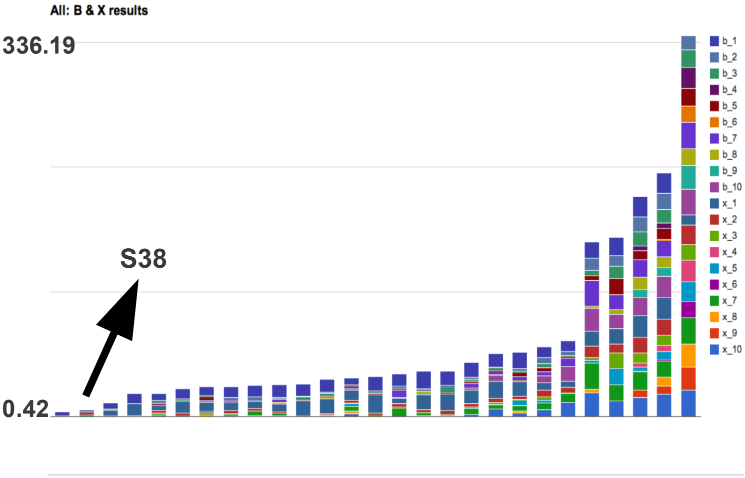
- 82 registered teams
- 48 teams sent a program for qualification
- 30 qualified teams
- 27 teams sent a program for final

Evaluation

- Let *S* be the solution of instance *I*, *B* be the best solution among competitors and *R* be the original reference solution

  Score(I) = (Cost(S) - Cost(B)) / Cost(R)
- The score of a team is sum of the scores of all instances

Competition: Final Results



Conclusions and Future Works

Conclusions

- MIP-based LNS approach is inferior for solving very large size problems with very limited time.
- CP-based LNS approach has good anytime-behaviour which is important when solutions must be reported subject to a time limit.
- Replenishing domains via incremental recomputation allows CP-based LNS approach to create and solve subproblems efficiently. This is a key-factor in finding good quality solutions in a limited time.

Future Works

- Exploit the multi-cores that might be available when solving the problem
- Automatic tuning of parameters
- Variants of the problem e.g., temporal constraints

Thank You.