Comparing Solution Methods for the Machine Reassignment Problem

Deepak Mehta, Barry O'Sullivan, and Helmut Simonis

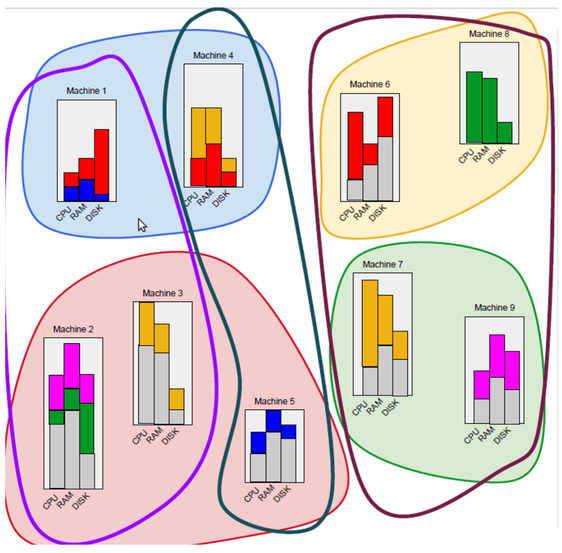
Fondúireacht Eolaíochta Éireann
Science Foundation Ireland

1 Machine Reassignment

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.

The **machine reassignment problem** is defined by a set of machines, M, and a set of processes, P.

- 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.
- A **service** is a set of processes. A set of services partition the set of processes. A **location** is a set of machines. A set of locations partition the set of machines. A **neighborhood** is a set of machines. A set of neighborhoods also partition the machines.



The task is to reassign the processes to machines while respecting a set of constraints in order to improve the utilisation of the machines, as defined by a complex cost function.

Constraints

- **Capacity**: The usage of a (transient) resource by a machine should not exceed its capacity.
- **Conflict**: The processes of a service should be assigned to different machines.
- **Spread**: The processes of a service should be spread over at least a given number of locations.
- **Neighborhood**: If a service s depends on another service s' then the neighborhoods of s must be a subset of the neighborhoods of s'.

Costs

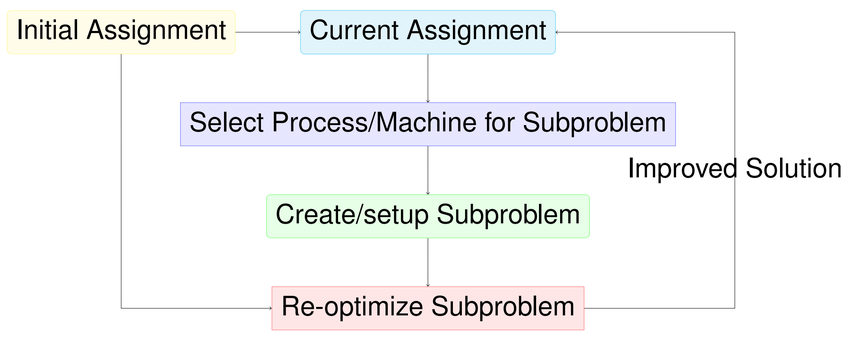
The objective is to minimize the weighted sum of the following costs:

- **Load**: Any usage above a given limit incurs a cost.
- **Balance**: To balance the availability of resources.
- **Process/Machine Move**: The cost of moving a process from its current machine to another machine.
- **Service Move**: To balance the movement of processes among services.

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

2 Large Neighborhood Search



Select Process/Machine for Subproblem

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

- Select $M' \subseteq M$, and then select $P' \subseteq P$ such that the currently assigned machine to each $p \in P'$ is in M'.
- **Closed subproblem:** For each $p \in P'$ the domain of machines is restricted to M'.
- **Open subproblem:** For each $p \in P'$ the domain of machines is set to M.

Create/setup Subproblem

1. Create the full problem model in memory. At each iteration reinitiaize the domains, reassign the machines to the processes which are not chosen for reassignment, and perform constraint propagation.
   - Space prohibitive depending upon the formulation.
   - Setting up the domains can be time consuming.
2. At each iteration create only a subproblem in memory by projecting the problem on the processes selected for reassignment and perform constraint propagation.
   - creating subproblem and setting up the domains at each iteration can be time consuming.
3. Create the full problem model in memory, and at each iteration unassign current machines from the selected processes, replinish the domains incrementally via recomputation.

Re-optimize subproblem

Branch and bound search with threshold on the number failures/time

2.1 MIP-LNS

- A set of machines, M', is selected by solving an optimization problem, where 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.
- Branch and Bound search with 10 seconds time-out.

2.2 CP-LNS

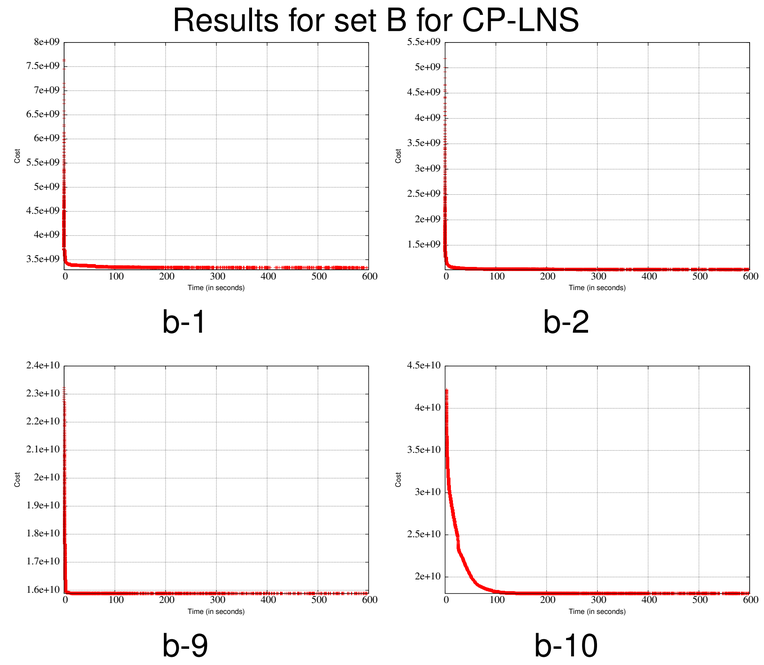
- M' is selected randomly, $1 \leq |M'| \leq 10$ and $|P'| \leq 40$.
- Full problem model is maintained. At each iteration the domains are replinished incrementally via recomputation.
- Branch and Bound with threshold on the number of failures, and cost-based heuristics are used to select process/machine during search.

3 Experimental Results

Results for set A obtained within 300 seconds

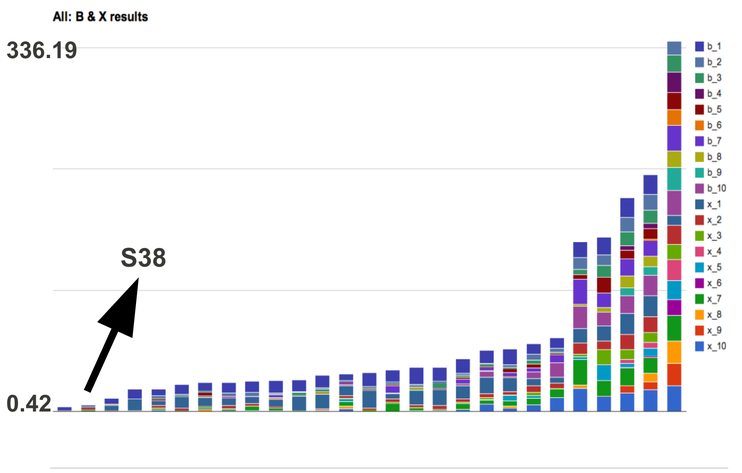
Prob	Initial	Best ROADEF	MIP-LNS	CP-LNS
a1-1	49,528,750	44,306,501	**44,306,501**	**44,306,501**
a1-2	1,061,649,570	777,532,896	792,813,766	**778,654,204**
a1-3	583,662,270	583,005,717	583,006,527	**583,005,829**
a1-4	632,499,600	252,728,589	258,135,474	**251,189,168**
a1-5	782,189,690	727,578,309	**727,578,310**	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**

Results for set B for CP-LNS



4 Competition Results

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



5 CONCLUSION

- MIP-LNS is inferior for very large size problems with very limited time.
- CP-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-LNS approach to create and solve subproblems efficiently.

Acknowledgements

This work has been supported by the Science Foundation Ireland under Grant No. 10/IN.1/I3032.

Cork Constraint Computation Centre, University College Cork, Ireland

eMail: {d.mehta|b.osullivan|h.simonis}@4c.ucc.ie