High-Accuracy Scalable Solutions to the Dynamic Facility Layout Problem

Apan Qasem[1], Clara Novoa[2], Chandra Kolla[1], Samantha Coyle[1]
[1]Department of Computer Science, [2] Ingram School of Engineering - Texas State University
cn17@txstate.edu, apan@txstate.edu, c_k35@txstate.edu , samc@txstate.edu

Abstract

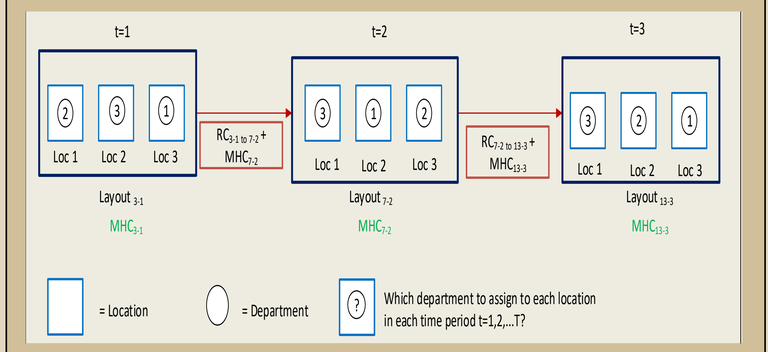
Between 20% and 50% of the operating expenses in manufacturing in the U.S are due to material handling and changes to plant layout. Effective facilities planning can reduce these costs by at least 10% to 30% and increase productivity.

The goal:
- Minimize total material handling cost (MHC)
- Minimize facilities relocation costs(RC)

Questions:
Which location to assign to a warehouse or a set of warehouses?
How to layout the products in a particular warehouse or set of warehouses?

We develop a parallel algorithm to solve the DFLP. Speedup for the best parallelized version vs. its serial counterpart is as high as 13. Experiments use 24 problems generated by [1] with 6, 15, and 30 facilities or departments. Accuracy of our solutions is higher than the ones in 3 out of 6 previous works. For 15 and 30 facilities, our solution is on average 3.46% above the best known solution (BKS). The worst case solution is 5.66% of the BKS.

The DFLP and a Non-Linear Programming Formulation



Decision variables:
x_{tki}: one if department k is assigned to location i in period t and zero otherwise
y_{tkiij}: one if department k is shifted from location i in period t − 1 to location j in period t and zero otherwise
Parameters:
T = Length of the planning horizon, usually given in years
n = Number of facilities and number of locations
f_{klt} = Flows of material between facilities k and l at time t
dij = Distances between any physical locations i
a_{tkij} = Cost of relocating department k from location i to j in period t

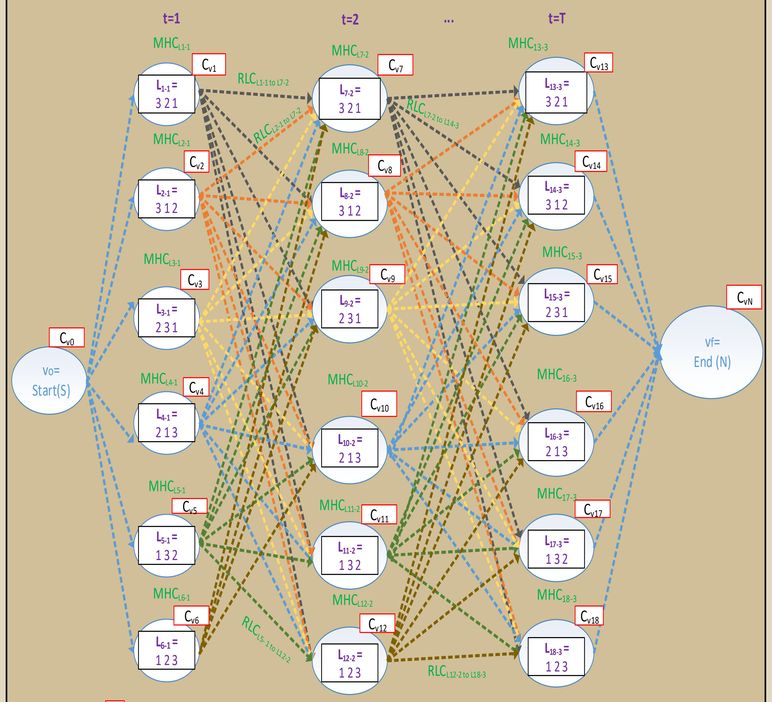
$$\min z = \sum_{t=1}^{T}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{tkl} d_{ij} x_{tki} x_{tlj} + \sum_{t=2}^{T}\sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{tkij}\, y_{tkij} \qquad (1)$$

$$s.t.$$

$$\sum_{i=1}^{n} x_{tki} = 1 \qquad t = 1,\dots,T; k = 1,2,\dots,n \qquad (2)$$

$$\sum_{k=1}^{n} x_{tki} = 1, \qquad t = 1,\dots,T; i = 1,2,\dots,n \qquad (3)$$

Network Representation



This network represents a very small problem with 3 departments and 3 time periods. All the 3! = 6 layouts are considered in each period. As the problem grows, including all the layouts in each period is not possible

Solution:
- Find all-pairs shortest path (SP) in the periodic layout graph
- Employ an augmented version of the Fredman-Tarjan algorithm (FTA) [2] to find the SP.

Observation:
- Selection of a node to label as marked does not require traversing the temporary labeled nodes to find the minimum cost. This is because there is no interest in identifying the nodes that are closer to the initial node v_0 in increasing order of cost
- Exploit to make FTA more efficient in practice

Algorithmic complexity:
- $(|E| + |V|\log|V|)$ (E = number of edges, V = Number of nodes)

Optimizations

MHC computation:
- Parallelize the |V| layouts by subdividing the layout space
- Assign a thread to tackle a whole
- Calculate MHC for a single layout
- Perform a parallel reduction across flows between each pair of facilities

RC computation:
- Parallelize across layouts for a given period

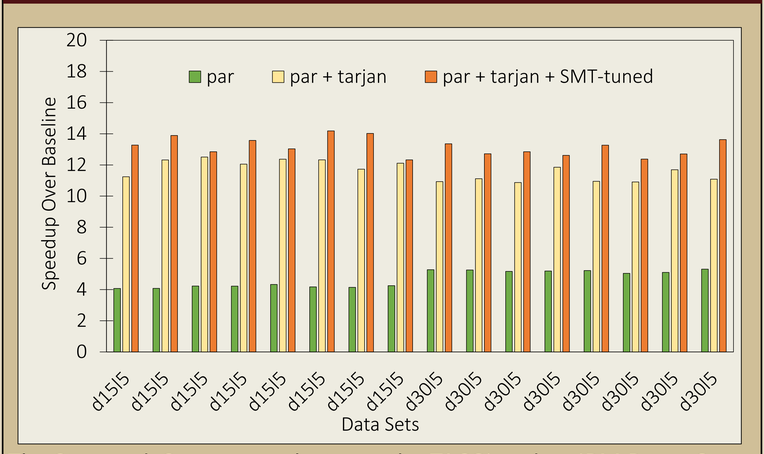
FTA cannot be parallelized across periods due to dependencies. The graph was partitioned based on configurations that belong to different periods and threads.

Time Skewing: RC and MHC computation were parallelized along the time dimension. We further apply a skewing optimization over the time-parallelized loop.

Random Re-start: Each instance is solved with a randomly selected subset of all feasible permutations (i.e. layout configurations). The process is repeated k times. The minimum cost for all k runs and parallelization methods tested becomes the best overall solution.

Sorting: Permutations generated are sorted and filtered based on the MHC to put in the network the best subsets of them.

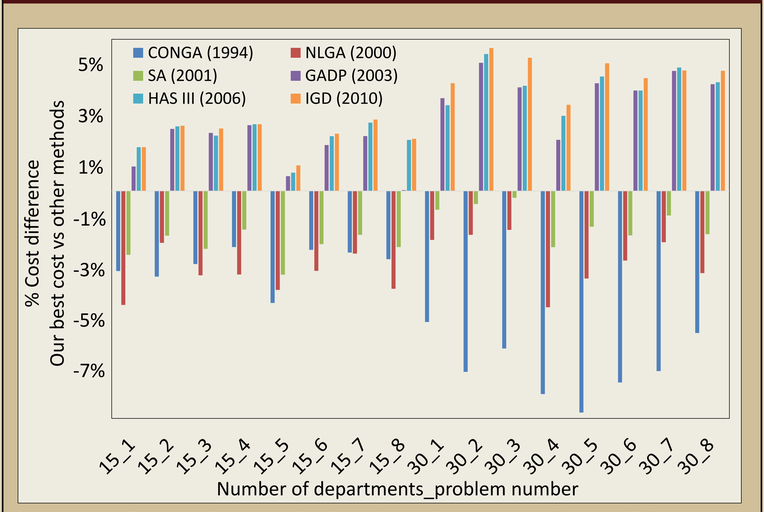
Performance



The Stampede2 compute cluster at the TACC+ and an IBM Power8 system were used to evaluate the solution. We developed an autotuning technique to select the optimal number of threads for each parallel segment in the application. Optimal thread affinity is also derived.

Maximum speed up factor is as high as 13 across all problems

Accuracy



Accuracy of our solution is higher than the ones in 3 out of 6 previous works. On average our solution is 3.46% above the best known solution.

Conclusion

Our parallel approach provides vast improvements in computational time over serial DFLP solutions and its best solutions are competitive if compared to existing methods using sophisticated meta-heuristics such as simulated annealing and genetic algorithms. On the Power8 system, the SMT-autotuning method is able to yield a further 16% performance improvement over the fully optimized parallel version.

Acknowledgements

This work was supported by NSF grants CNS-1253292 and OAC-1829644 and an equipment donation by IBM. Texas Advanced Computing Center (TACC) at The University of Texas at Austin also providing HPC resources that have contributed to the research results reported within this paper.