Barbara Sudol bas334, Maxwell Wulff mcw232
December 2018
ORIE 4330 Final Project Part IV Report

For our infeasible search algorithm we consider the best possible swap that is allowed to violate the capacity constraint and the conflict constraint. At a high level the code:

1. Performs a local search
2. Finds the best swap it hasn't looked at before
3. Attempts the swap
4. Attempts to solve the infeasibility
5. Keeps the swap and the fix if it exists and doesn't raise the price

Some considerations we had to make:

1. Separating between capacity infeasibility and conflict infeasibility
2. The fact that it is possible that no swap improves the cost
3. The fact that sometimes an infeasible swap cannot be fixed

Unfortunately our program takes a very long time to run, so we only ran it for instance 5. However the bound that we compute is good, reasonably close to the lower bound computed.

In order to compute a lower bound on the optimal value for the problem, we implemented a linear program relaxation of the problem using the gurobi python solver. We added decision binary variables X_{ij} for all machines i and processes j, equaling 1 if process j was assigned to machine i and 0 otherwise. With these variables in mind, we wrote constraints corresponding to the capacity constraints, conflict constraints and balance constraints. These were the only constraints we needed to implement for the LP to return good lower bounds (always lower than our local minimum values).

One issue we found with testing the local search on the sample instances is runtime: as our method considered every possible pair of processes which can grow extremely large. We altered our local search method to consider every 20th pair if the number of processes was over 500 to make it run in a reasonable amount of time and the costs were still an improvement over original costs, and in most instances fairly close to our lower bound.
However, our gurobi LP relaxation was able to compute lower bounds on these problems in significantly less time than our local search method.
Below: results from running our code on the provided sample instances:

Instance	Original Cost	After Local Search	After Solve	Gurobi Lower Bound
a1_1	49528750	44306602	Really Slow	44306389
a1_2	1061649570	1054605172	Really Slow	777530729
a1_3	583662270	583523023	Really Slow	583005700.0

a1_4	632499600	591254698	Really Slow	242387530.0
a1_5	782189690	729288072	727581225	727578289