Integer Programming
Practical Project

Franks, Billy Joe
b_franks12@cs.uni-kl.de

Renger, Justus
jus.renger@gmail.com

February 13, 2019

This document will focus on explaining the intricacies of a specific implementation to solve Minimum Row Misses and will go into achieved results. This document is part of a Practical Project done for the course Integer Programming at the Technische Universität Kaiserslautern.

1 Methodology

Our code for solving Minimum Row Misses is mostly based on randomized algorithms. There are two subproblems we consider, one is the subproblem assuming the $b := 1$, further referred to as MRMb, the second assumes $c := 1$, further referred to as MRMc.

The Idea will be to first generate a base of feasible solutions, this is an easy task explained further later on, and then to iteratively refine all of these solutions. The refinement is based on MRMb and MRMc, we first try to refine each bank separately using MRMb on each bank and the subsequence obtained from ignoring all elements not contained in this bank. Then we collapse each row into one element representing this row and try to find an allocation into banks using MRMc. This is repeated until neither MRMb nor MRMc can find an improvement, in this sense this is a greedy algorithm.

1.1 Feasible Solution

Feasible solution can be obtained by just allocating the elements randomly in any order into the banks and their rows, whenever allocating to a row which is full simply reallocate the element.

Our methodology is a bit different but effectively similar. We go through the sequence and collect each element in order of first appearance. Using this array we can fill the first bank, ignoring how many rows are allowed, by filling the first row, then the second and so on. This gives a ”good guess”, which typically performed quite well, we call this solution the trivial solution. To bring randomness into the picture we can take the list of elements and shuffle them and then repeat the procedure, we will call these types of solutions randomRows. To bring even more initializations into the picture we can fill columns first instead of rows, we will call these solution randomColumns. This brings potential benefits related to the later methods MRMb and MRMc.

Our final submission first refines the trivial solution and then iteratively refines a randomRows and then a randomColumns solution. Since we do not have any kind of termination condition, we will simply give the algorithm the maximum amount of time it is allowed to search.(Due to some reading and writing constraints this time is only regarded within +- 10 seconds)

1.2 MRMb

Assuming $b := 1$ we take two rows at random and try to exchange elements to improve the amount of misses. This is done using a minimum cut which respects an amount of elements per set. More specifically the sequence has to be divided into a set of sequenced of consecutive elements contained in the two rows. We can then construct the misses produces by putting two element into different sets similarly to how it was done for k-cut. This gives us a minimum-cut problem. We randomly shuffle the elements into both sets of the cut and then search for improvement in a greedy fashion, checking wether the amount of misses can be reduced by exchanging or moving elements to a different set. If the graph produced by the previous method has different components we only apply this cutting strategy if one of the components is larger than one row. If not we use binpacking to pack the components into the two sets, more specifically if our bank has empty rows we use 3 bins instead of 2. If after binpacking one bin is too full we use the greedy approach to exchange elements again.

We dont choose the two rows randomly but instead for each pair of rows put them into a list a few times and then shuffle this list and work through this list, to be sure to have compared each pair of rows if time permits.

1.3 MRMc

Assuming $c := 1$ we take two random banks and try to exchange elements between the banks to improve the amount of misses. This is done by shuffling the elements into two sets and checking wether the amount of misses has improved.

We dont choose the two columns randomly but instead do the same as for MRMb and its rows.

2 Results

The time for all experiments was 5 minutes or 300 seconds, with deviations +-10 seconds, especially for the last sequence. 131072_393216.seq was infeasible for all settings so it will be omitted. The table it separated into 3 part for $b \in \{1, 2, 4\}$ denoted by the number besides misses of hits.

Sequence	Misses1	Hits1	Misses2	Hits2	Misses4	Hits4
10_50.seq	25	25	7	43	3	47
99_500.seq	367	133	323	177	246	254
190_950.seq	738	212	634	316	493	457
278_1400.seq	1138	262	1024	376	877	523
546_2750.seq	2327	423	2080	670	1702	1048
792_19024.seq	1366	17658	1279	17745	1125	17899
4224_8402.seq	132	8270	132	8270	132	8270
6121_12155.seq	192	11963	192	11963	192	11963
10336_453358.seq	8970	444388	8806	444552	8447	444911
10400_294960.seq	5762	289198	5618	289342	5398	289562
11296_316796.seq	5892	310904	5738	311058	5531	311265
11520_644456.seq	12833	631623	12397	632059	11966	632490
356400_1198800.seq	93192	1105608	53541	1145259	26385	1172415