Dynamic Binary Translation (DBT) has been used as an approach to transparently run code on different architectures and has generally made use of runtime information to perform effective dynamic compiler optimization. Dynamic optimization techniques are hard to design, as they have to improve code performance under stringent runtime constraints. In this paper we present preliminary work on a code optimization technique called Hole Allocation. Our goal with this paper is to introduce and highlight its potential, and to point new directions for improvement. Hole Allocation uses runtime information, collected by DBTs, to identify free register ranges (holes) which could be target of register promotion. Preliminary experiments conducted with the SPEC CPU2000 benchmark, using only one memory access pattern, shows that Hole Allocation can achieve, for some program runs, moderate speedup-ups, but also reveal that performance can suffer in some cases, and needs to be addressed.