Implementation of parallel optimization algorithms using generalized branch and bound template

M.Baravykaitė, VGTU
mmb@fm.vtu.lt

J.Žilinskas, MII
julius.zilinskas@mii.lt
Content

- Goals
- Branch and bound (BB) algorithms
- BB algorithm template
- Some results of experiments
What is considered

- Applications of optimization
  - Solve optimization problem for some practical application
    - Implemented optimization algorithm

- New algorithms
  - Testing of new optimization algorithms
    - Some tests, standard methods
BB algorithm

- Consider minimization problem
  \[ \min_{x \in S} f(x) \]

- Main rules:
  - Covering rule
  - Subspace selection rule
  - Branching rule
    - Divide search space into subspaces
  - Bounding and elimination
    - Find out subspaces that does not contain optimal solution and eliminate them form further search
Generalized BB algorithm

Cover solution space $D$ by $L = \{L_j | D \subseteq \bigcup L_j, j = 1, m\}$ using covering rule. $S = \emptyset$, $UB(D) = \infty$.

while sub-space list is not empty $L \neq \emptyset$ do

Choose $I \in L$ using selection rule, exclude $I$ from $L$.

if $LB(I) < UB(D) + \epsilon$ then

Branch $I$ into $p$ subsets $I_j$ using branching rule.

for all $I_j, j = 1, p$ do

Find $UB(I_j \cap D)$ and $LB(I_j)$ using bounding rules.

$UB(D) = \min(UB(D), UB(I_j \cap D))$.

if $LB(I_j) < UB(D) + \epsilon$ then

if $I_j$ is a leaf then $S = I_j$.

else $L = \{L, I_j\}$.

end if

end if

end for

end if

end while
Template based programming

- Separate problem dependent part from the general structure of the algorithm.
- Implement general structure of the algorithm and use it to solve different problems.
BB template class scheme
Parallel BB algorithm

- Any subspace of initial space can be searched independently and in any order

Parallel BB algorithm:
- Divide initial space into subspaces
- Distribute subspaces among the processors
  - Subspaces can be redistributed later for better load balancing
- BB algorithm is asynchronous performed over the mapped subspace and best function value is exchanged.
- At the end, the best solution is chosen among the processors
Parallel BB specifics

- Different search order of the subspaces
- Search overhead factor

\[
sof = \frac{W_p}{W}
\]

- \( W \) – amount of work done by a single processor,
- \( W_p \) – total amount of work done by \( p \) processors

- Processor load disbalance
- Communications
Experiments

Test problems
- Travelling salesman problem over 20 cities
- Lipshitz functions with known Lipshitz constant

Experiments were performed on VGTU cluster Vilkas (up to 10 processors) and IDRIS supercomputing center (up to 256 processors)

Speedup \( S_p = \frac{T}{T_p} \)
Experiments (Lipshitz function)
Experiments (Lipshitz function)
Processor load disbalance
Creation of new algorithms

- Possibility to construct new algorithms using BB algorithm template by specifying bounding, branching and other rules.
- New algorithm can be tested using implemented search orders and problem methods.
Final remarks

- BB algorithm template can be used choose the suitable sequential BB algorithm
- BB algorithm template helps obtain parallel programs easily.
- New algorithms can be tested using prepared template functions used for BB.