Variable Neighbourhood Decomposition Search for Mixed Integer Programs

Jasmina Lazić[1], *Nenad Mladenović*[1], *Dragan Urošević*[2]

[1] Brunel University, West London, UK

[2] Mathematical Institute, Serbian Academy of Sciences, Belgrade, Serbia

Presentation Outline

- Introduction and motivation
- Relaxation Induced Neighbourhood Search (RINS)
- Variable Neighbourhood Decomposition Search (VNDS)
- Local search within VNDS (Variable Neighbourhood Descent)
- Computational Results
- Conclusion

Mixed Integer Program (MIP)

General mixed integer program can be formally defined as:

$$
\begin{aligned}
(P) \qquad & \min c^{\mathrm{T}} x && && (1)\\
& Ax \geq b && && (2)\\
& x_j \in \{0,1\} && \forall j \in \mathcal{B} \neq \emptyset && (3)\\
& x_j \geq 0, x_j \in \mathbb{Z} && \forall j \in \mathcal{G} \neq \emptyset && (4)\\
& x_j \geq 0 && \forall j \in \mathcal{C} \neq \emptyset && (5)
\end{aligned}
$$

where set of indices $N = \{1, 2, \ldots, n\}$ is partitioned into three subsets $(\mathcal{B}, \mathcal{G}, \mathcal{C})$, corresponding to binary, general integer and continuous variables, respectively.

Solving MIPs

- MIPs are typically solved by using branch-and-bound or branch-and-cut algorithm.
- A number of models is hard to solve to optimality only by branch-and-bound/cut.
- This suggests applying local search techniques (introducing neighbourhoods, intensification and diversification steps).

Integrality vs. Optimality

- Incumbent solution: feasible with respect to integrality constraints, but not optimal unless it is the last and optimal integral solution.
- Linear relaxation solution: has the best possible objective value, but is usually not feasible with respect to integrality constraints.

Relaxation Induced Neighbourhood Search (RINS)

- Recent metaheuristic proposed by Emilie Danna, Edward Rothberg and Claude Le Pape from ILOG in 2004.
- Based on the fact that many variables in the incumbent and in the continuous relaxation solution have the same values.
- Focuses on the variables that differ in the incumbent and in the continuous relaxation at the current node.

Outline of the RINS Algorithm

At a node of the global branch-and-cut tree, perform the following steps:

1. Fix the variables that have the same values in the incumbent and in the current continuous relaxation;
2. Set an objective cutoff based on the objective value of the current incumbent;
3. Solve a sub-MIP on the remaining variables.

Variable Neighbourhood Search

- Metaheuristic proposed by N. Mladenović and P. Hansen in *Computers Oper. Res.* 24: 1097–1100, 1997.)
- Based on the systematic change of neighbourhoods within a local search.

Neighbourhoods in Problem Solution Space

- Let d denote the distance metric in the solution space X of the problem observed.
- The k-th neighbourhood of solution $x \in X$ is usually defined as the following set:

$$\mathcal{N}_k(x) = \{y \in X \mid d(x, y) \leq k\},$$

Basic VNS

Initialisation.

(1) Select parameters k_{min}, k_{max} and k_{step} for neighbourhood exploration.

(2) Select the set of neighbourhood structures $\mathcal{N}_k$, for $k = k_{min}, k_{min} + k_{step}, \ldots, k_{max}$.

(3) Find an initial solution x.

(4) Choose a stopping condition.

Main step.

Repeat until the stopping condition is met:

(1) Set $k \leftarrow k_{min}$.

(2) **Repeat** until $k = k_{max}$:

 (a) *Shaking.* Generate $x' \in \mathcal{N}_k(x)$ at random.

 (b) *Local search.* Apply some local search with x' as initial solution. Denote with x'' so obtained local optimum.

 (c) *Move or not.* If x'' is better then x, move there (set $x \leftarrow x''$) and continue the search in $\mathcal{N}_1(x)$. Otherwise set $k \leftarrow k + k_{step}$.

Variable Neighbourhood Descent (VND)

Initialisation.

(1) Select maximal neighbourhood size k_{max}.

(2) Select the set of neighbourhood structures $\mathcal{N}_k$, for $k = 1, 2, \ldots, k_{max}$.

(3) Find an initial solution x.

Main step.

Repeat until the no improvement is obtained:

(1) Set $k \leftarrow 1$.

(2) **Repeat** until $k = k_{max}$:

 (a) *Neighbourhood exploration.* Find the best neighbour x' of x, $x' \in \mathcal{N}_k(x)$.

 (c) *Move or not.* If x' is better then x, move there (set $x \leftarrow x'$) and continue the search in $\mathcal{N}_1(x)$. Otherwise set $k \leftarrow k + 1$.

Variable Neighbourhood Decomposition Search (VNDS)

- Basic VNS remains difficult or long to solve very large instances of problems.
- VNDS extends the basic VNS scheme into two-level VNS scheme based upon the decomposition of the problem.

Outline of the VNDS Algorithm

Initialisation.

(1) Set parameters k_{min}, k_{max} and k_{step}.

(2) Select the set of neighbourhood structures $\mathcal{N}_k$.

(3) Find an initial solution x.

(4) Choose a stopping condition.

Main step.

Repeat until the stopping condition is met:

(1) Set $k \leftarrow k_{min}$.

(2) **Repeat** until $k = k_{max}$:

- (a) *Shaking.* Generate $x' \in \mathcal{N}_k(x)$ at random. Let $y = x' \backslash x$.
- (b) *Local search in problem subspace.* Apply some local search with y as initial solution. Denote with y' so obtained local optimum and with $x'' = (x' \backslash y) \cup y'$.
- (c) *Local search in the whole problem space.* If x'' is better than x, apply some local search with x'' as initial solution and denote with x''' so obtained local optimum. Otherwise set $x''' \leftarrow x''$.
- (d) *Move or not.* If x''' is better then x, move there (set $x \leftarrow x'''$) and continue the search in $\mathcal{N}_1(x)$. Otherwise set $k \leftarrow k + k_{step}$.

VNDS for MIPs

- Based on the idea first explored in RINS: that many variables in the incumbent and in the continuous relaxation values have the same values.
- Number of variables having the same values in both solutions is used to define the decomposition scheme within VNDS.

VNDS Algorithm for MIPs: Initialisation

I1) Find the continuous relaxation solution y.

I2) Find the first feasible mixed integer solution $\mathbf{x} = (x_1, x_2, \ldots, x_m, x_{m+1}, \ldots, x_n)$, $x_1, \ldots, x_m \in \mathbb{Z}$, $x_{m+1}, \ldots, x_n \in \mathbb{R}$. Denote with f_{opt} objective function value.

I3) Set values for t_{max} (total running time allowed) and t_{sub} (time allowed for solving subproblem). Set $t_{start} = cpuTime(\,)$, $t = 0$.

Main Step - Decomposition Initialisation

M1) Reorder $\mathbf{x}$: Set $\mathbf{x} = (x_{i_1}, x_{i_2}, \ldots, x_{i_m}, x_{m+1}, \ldots, x_n)$, so that $\mid x_{i_1} - y_{i_1} \mid \leq \mid x_{i_2} - y_{i_2} \mid \leq \ldots \leq \mid x_{i_m} - y_{i_m} \mid$.

M2) Set $\mathbf{z} = (z_1, z_2, \ldots z_m, z_{m+1}, \ldots, z_n) = (x_{i_1}, x_{i_2}, \ldots, x_{i_m}, x_{m+1}, \ldots, x_n)$.

M3) $totDiff = m - \max\{k \in \{1, \ldots, m\} \mid x_{i_k} - y_{i_k} = 0\}$, $k_{min} = [totDiff/10]$, $k_{step} = k_{min}$, $k_{max} = m$, $k = k_{min}$.

M4) Fix values of $z_1, z_2, \ldots, z_{k_{max}-k_{min}}$. Set $upperBound = f_{opt} - 0.00001$.

Main Step - Decomposition

M5) **Repeat** until ($k \geq k_{max}$ or $t \geq t_{max}$)

D1) MIPSOLVE(t_{sub}, $upperBound$, $\mathbf{z}_{next}$, f_{next})

D2) <u>*Move or not.*</u>

if ($f_{next} < f_{opt}$) **then**

a) <u>*Relax.*</u> Relax all fixed variables from $\mathbf{z}$.

b) <u>*Local search in the entire problem space.*</u>

VND(t_{sub}, $upperBound$, $\mathbf{z}_{cur}$, f_{cur}).

c) <u>*Move.*</u> Set $\mathbf{x} = \mathbf{z}_{cur}$, $f_{opt} = f_{cur}$. Go to step M1).

else

if ($k + k_{step} > totDiff$) **then**

$k_{step} = \max\{[(k_{max} - k)/2], 1\}$

endif

endif

D3) Relax variables $z_{k_{max}-k-1}, z_{k_{max}-k-2}, \ldots, z_{k_{max}-k-k_{step}}$.

D4) $k = k + k_{step}$, $t_{end} = cpuTime(\,)$, $t = t_{end} - t_{start}$.

Entire Space Local Search (part I): Distance in the MIP Solution Space

- Given two feasible solutions x and y of (P) we can define distance between them as

$$\Delta(x,y) = \sum_{j \in \mathcal{B} \cup \mathcal{G}} |\, x_j - y_j \,|$$

- It is easy to see that, for binary MIPs, this distance is identical to *Hamming distance*:

$$\Delta(x,y) = \sum_{j \in \overline{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \overline{S}} x_j,$$

where $\overline{S} = \{j \in \mathcal{B} \mid y_j = 1\}$.

Entire Space Local Search (part II): Neighbourhood Structures for Binary MIPs

- Let (P) be a binary MIP ($\mathcal{G} = \emptyset$) and Δ previously defined distance in the solution space X of (P)
- The kth neighbourhood of any feasible solution x of (P) is defined as:

$$\mathcal{N}_k(x) = \{y \in X \mid \Delta(x, y) \leq k\},$$

i.e.

$$\mathcal{N}_k(x) = \{y \in X \mid \sum_{j \in \overline{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \overline{S}} x_j \leq k\}$$

- $\mathcal{N}_k(x)$ is obviously the set of all solutions of (P), which differ from x in at most k binary variables.

Entire Space Local Search (III): VND for Binary MIPs

(1) *Initialisation.* Set $proceed \leftarrow$`true`, $rhs \leftarrow 1$, $first \leftarrow$`false`.

(2) **while** ($proceed$ or $elapsedTime < totalTimeLimit$) **do**

$timeAllowed \leftarrow \min(nodeTimeLimit, totalTimeLimit - elapsedTime)$;

Add local branching constraint $\Delta(x, x_{cur}) \leq rhs$;

Set $upperBound \leftarrow f_{cur}$;

Set $status \leftarrow$ MIPSOLVE($timeAllowed$, $upperBound$, $first$, x_{next}, f_{next});

switch $status$ **do**

case "optSolFound": reverse last local branching constraint into

$\Delta(x, x\_cur) \geq rhs + 1$; $x_{cur} \leftarrow x_{next}$; $f_{cur} \leftarrow f_{next}$; $rhs \leftarrow 1$;

case "feasibleSolFound":reverse last local branching constraint into

$\Delta(x, x\_cur) \geq 1$; $x_{cur} \leftarrow x_{next}$; $f_{cur} \leftarrow f_{next}$; $rhs \leftarrow 1$;

case "provenInfeasible":reverse last local branching constraint into

$\Delta(x, x\_cur) \geq rhs + 1$; $rhs \leftarrow rhs + 1$;

case "noFeasibleSolFound": $proceed \leftarrow$ `false`;

end

end

Results

- Our code is written in C++ and built in Microsoft Visual Studio 2005 under Windows XP operating system.
- All experiments are run on Pentium 6 computer with 2.4GHz processor and 4GB RAM.
- The data sets used are:
  - 7 MIPLIB-3.0 instances `mkc`, `swath`, `danoint`, `markshare1`, `markshare2`, `arki001` and `seymour`
  - 1 network design instance `net12`
  - 2 crew scheduling instances `biella1` and `NSR8K`
  - 1 railway crew scheduling instance `rail507`
  - 1 nesting instance `glass4`
  - 2 telecommunication network design instances `UMTS` and `van`
  - 2 rolling stock and line planning instances `roll3000` and `nsrand_ipx`
  - 5 lot-sizing instances `A1C1S1`, `A2C1S1`, `B1C1S1`, `B2C1S1` and `tr12-30`
  - 4 railway line planning instances `sp97ar`, `sp97ic`, `sp98ar` and `sp98ic`

  which is the standard benchmark for testing binary MIP solvers.

Table of Objective Values

Model	Best published	VNDS	CPLEX without RINS	RINS
mkc	-563.85	-561.94	-563.85	-563.85
swath	467.41	480.12	509.56	524.19
danoint	65.67	65.67	65.67	65.67
markshare1	7.00	***3.00***	5.00	7.00
markshare2	14.00	***10.00***	15.00	17.00
arki001	7580813.05	7580814.51	7581076.31	7581007.53
seymour	423.00	425.00	424.00	424.00
NSR8K	20780430.00	***20763500.00***	163138974.32	93373309.04
rail507	174.00	174.00	174.00	174.00
glass4	1400013666.50	1587513455.18	1575013900.00	1460007793.59
van	4.84	5.09	5.35	5.09
biella1	3065084.57	***3065005.78***	3065729.05	3071693.28
UMTS	30122200.00	30125601.00	30133691.00	30122984.02
net12	214.00	214.00	255.00	214.00
roll3000	12890.00	12930.00	12890.00	12899.00
nsrand_ipx	51360.00	***51200.00***	51360.00	51360.00
a1c1s1	11551.19	***11503.44***	11505.43	***11503.44***
a2c1s1	10889.14	10958.42	10889.14	10889.14
b1c1s1	24544.25	24646.77	24908.63	24544.25
b2c1s1	25740.15	25997.84	25869.40	25740.15
tr12-30	130596.00	130596.00	130596.00	130596.00
sp97ar	662671913.92	665917871.36	670484585.92	662892981.12
sp97ic	429562635.68	***429129747.04***	437946706.56	430623976.96
sp98ar	529814784.70	531080972.48	536738808.48	530806545.28
sp98ic	449144758.40	451020452.48	454532032.48	449468491.84

Table of Gap Values (in %)

Model	VNDS	CPLEX without RINS	RINS
mkc	0.34	0.00	0.00
swath	2.72	9.02	12.15
danoint	0.00	0.00	0.00
markshare1	0.00	66.67	133.33
markshare2	0.00	50.00	70.00
arki001	0.00	0.00	0.00
seymour	0.47	0.24	0.24
NSR8K	0.00	685.70	349.70
rail507	0.00	0.00	0.00
glass4	13.39	12.50	4.29
van	5.17	10.60	5.13
biella1	0.00	0.02	0.22
UMTS	0.01	0.04	0.00
net12	0.00	19.16	0.00
roll3000	0.31	0.00	0.07
nsrand_ipx	0.00	0.31	0.31
a1c1s1	0.00	0.02	0.00
a2c1s1	0.64	0.00	0.00
b1c1s1	0.42	1.48	0.00
b2c1s1	1.00	0.50	0.00
tr12-30	0.00	0.00	0.00
sp97ar	0.49	1.18	0.03
sp97ic	0.00	2.05	0.35
sp98ar	0.24	1.31	0.19
sp98ic	0.42	1.20	0.07
average:	1.02	34.48	23.04

Table of Running Times (in seconds)

Model	VNDS	CPLEX without RINS	RINS
mkc	9003.01	18000.47	18000.53
swath	3176.81	1283.23	557.93
danoint	3360.01	18000.63	18000.66
markshare1	370.81	10018.84	18000.58
markshare2	15448.03	3108.12	7294.20
arki001	4684.51	338.56	27.03
seymour	9150.68	18000.59	18000.69
NSR8K	34601.77	36001.68	36001.48
rail507	1524.28	662.26	525.25
glass4	625.44	3732.31	4257.54
van	1331.75	18001.10	18959.00
biella1	4452.02	18000.71	18000.61
UMTS	6836.78	18000.75	18000.59
net12	129.85	18000.75	18000.64
roll3000	2585.47	18000.86	14193.31
nsrand_ipx	10595.37	13009.09	11286.30
a1c1s1	1437.73	18007.55	18000.64
a2c1s1	2357.14	18006.50	18002.35
b1c1s1	5346.82	18000.54	18000.64
b2c1s1	133.19	18003.44	18000.80
tr12-30	1581.02	7309.60	4341.23
sp97ar	18025.51	11841.78	8498.12
sp97ic	3084.88	1244.91	734.96
sp98ar	4367.67	1419.13	1051.53
sp98ic	676.43	1278.13	1031.02
average:	5795.48	12290.86	12270.71

Conclusion

- We have managed to improve effectiveness on the largest instances such as `NSR8K` and `biella1`, proving that decomposition is successful in tackling the large problems.
- In the overall, we have improved the best known results in 7 out of 25 cases.
- Apart from being successful in solving large instances, VNDS has also proved to achieve the best results for instances `markshare1` and `markshare2`, belonging to the class of hard small 0-1 problems.
- VNDS significantly decreases the running time performance: its average running time is only 5795.48s, while the average time of RINS is 12270.71s, and of CPLEX without RINS 12290.86s.
- VNDS also sustains much higher stability than RINS or CPLEX without RINS: its average gap is only 1.02%, as opposed to 23.04% RINS gap and 34.48% gap of CPLEX without RINS.
- Future work: more improvement could be expected if we applied VNDS method at the other nodes of branch-and-bound tree and not just at the root node.