Neighborhood Management in Variable Intensity Neighborhood Search

Tatjana Davidović[1], Spomenka Milić[2]

[1]Mathematical Institute,
Serbian Academy of Sciences and Arts;
[2]Faculty of Mathematics, University of Belgrade

Jan. 27, 2023

SMSCG 2023

Presentation outline

1. Matheuristics
2. Variable Intensity Neighborhood Search
3. Neighborhoods Influence Analysis
4. Conclusion

What are Matheuristics?

- The name is compound of mathematical heuristics (*math-heuristic*)
- They are hybrids between metaheuristic methods and MILP solvers
- Metaheuristics are used to create subproblems that are treated by exact MIP solvers
- The size of subproblems is growing as the algorithm proceeds the execution
- In general, matheuristic are exact methods: if provided enough resources (time and memory), the subproblem becomes the original problem

Examples of Matheuristics

- HyperOpt, Burke et al. 2001
- Local Branching (LB), Fischetti & Lodi, 2003
- Variable Neighborhood Branching (VNB), Hansen *et al.*, 2006
- MetaBoosting, Pushinger et al. 2009
- MIP-Based GRASP, Dolgui et al. 2009
- Kernel Search, Enrico, Mansini, Speranca, 2010

Examples of Matheuristics (cont.)

- Mitrović-Minić, S., Punnen, A. P., Local search intensified: Very large-scale variable neighborhood search for the multi-resource generalized assignment problem, *Discrete Optimization*, 6(4), 370–377, 2009.

- Lazić, J., Hanafi, S., Mladenović, N., Urošević, D., Variable neighbourhood decomposition search for 0–1 mixed integer programs, *Comput. Oper. Res.*, 37(6), 1055–1067, 2010.

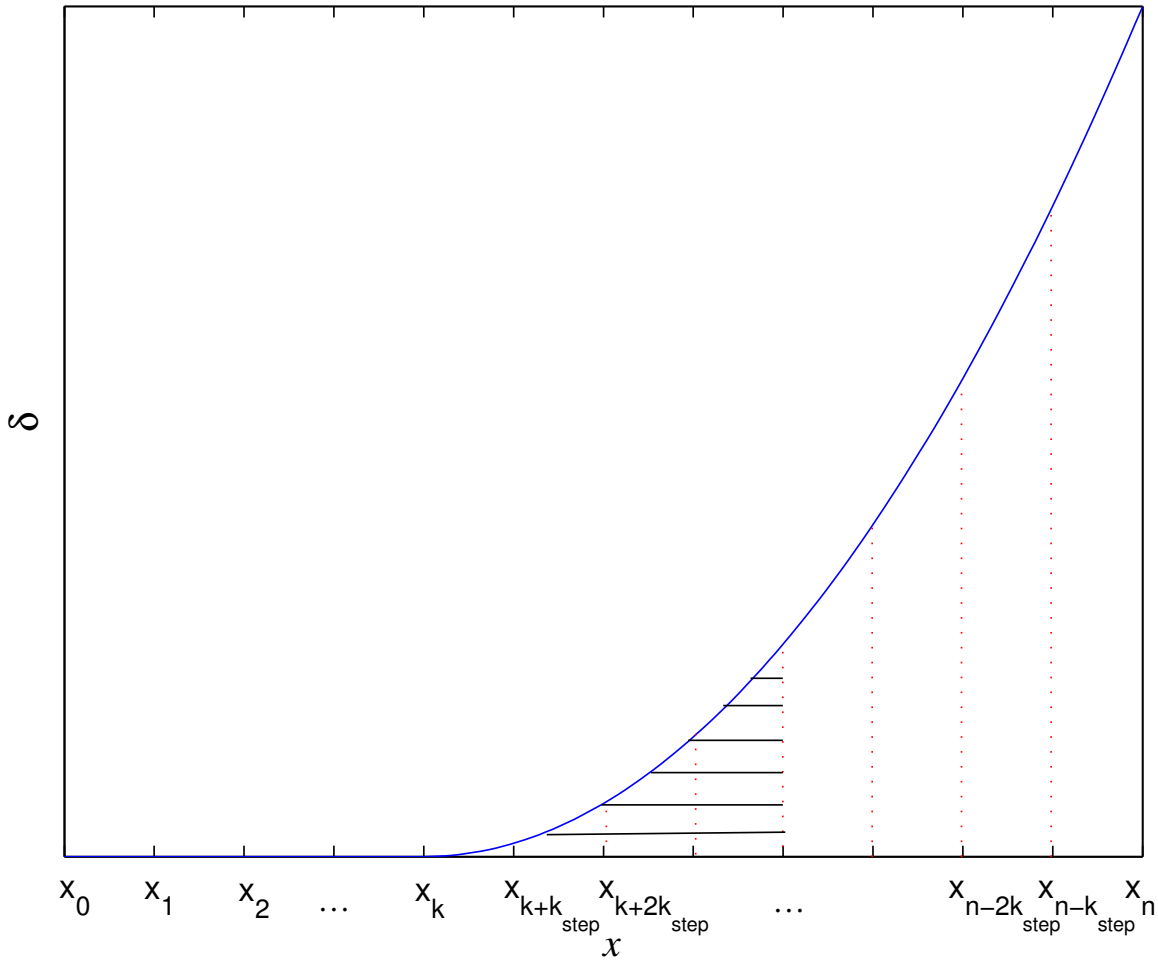
These two were the inspiration to develop Variable Intensity Neighborhoods Search (VINS)

- Jovanović, P., Davidović, T., Lazić, J., Mitrović-Minić, S., The variable intensity neighborhood search for 0-1 MIP, In *Proc. 42nd Symposium on Operations Research*, SYM-OP-IS 2015, p. 229–232, Srebrno jezero, Serbia, Sept. 15-18, 2015.

VINS - The main ideas

- Patterns for subproblem generation, obtained by fixing and relaxing subsets of problem variables, are similar to the ones in VNDS-MIP
- Neighborhood definitions are inspired by VILS
- At the beginning, linear relaxation solution and first feasible solution are found
- Variables are sorted in the non-increasing order with respect to the distance between these two solutions
- Subsets of variables are fixed to their current values depending on the current neighborhood
- The remaining variables are determined by the exact MILP solver

Illustration of subproblem generation



VINS - Neighborhoods definition

- N1: α% of the worst variables are released
- N2: variable set is divided into 10 bins and $\alpha/10$% worst variables are released within each bin
- N3: starting at random position α% variables are released
- N4: at 10 random positions, $\alpha/10$% variables are released
- N5: α% of the best variables are released

VINS - Neighborhoods definition (cont.)

- N6: variable set is divided into 10 bins and $\alpha/10\%$ best variables are released within each bin
- N7: $\alpha/2\%$ of best and worst variables are released
- N8: within 10 equal bins the same pattern as in N7 is applied
- N9: random $\alpha\%$ variables are released
- N10: in 10 equal bins, random $\alpha/10\%$ variables are released

VINS - Pseudo-code

Initialization: Read *MIP*, *Tlim*, *alphas*, *time_limits*;
    Solve linear relaxation of *MIP* to obtain *linx*;
    Calculate x, the first feasible solution for *MIP*;
    $improvement \leftarrow true$;
While $(t < Tlim)$
    If $(improvement)$
        $bestx \leftarrow x$;
        Add objective constraint $f(x) < f(bestx)$;
        Sort variables according to $(bestx - linx)$;
        $improvement \leftarrow false$;
    End If
    $N \leftarrow NextNeighborhood$;
    If $(N = null)$
        $N = N1$;
        $\alpha \leftarrow NextAlpha$;
        If $(\alpha = null)$
            $\alpha = 100$;
            break;
        End If
    End If
    $Release(N, \alpha)$;
    $x \leftarrow MIPSolve(MIP, TL)$;
End While

Experimental settings

- VINS is coded in Python 3.8
- Executed on Intel Core i7-2600 CPU 3.40GHz, 12 GB RAM, 930 GB SATA under Linux 5.14.6
- Test examples (26 instances):
  1. MIPLIB 3.0 - 9 instances
  2. AVRP - 2 instances
  3. Ship routing I model - 10 instances
  4. Ship routing II model - 5 instances

Parameter settings

Stopping criterion: $t_{lim} = 3600$ sec.

Table: Values for VINS parameters

n_{sizes}	α_1	α_2	α_3	t_{lim1}	t_{lim2}	t_{lim1}
3	20	40	60	$t_{lim}/10$ 360	$2\, t_{lim}/10$ 720	$2.5\, t_{lim}/10$ 900

VINS results

Inst.	#var(#bin.)	#constr.	Obj.	min_t
10teams	2025(1800)	230	924.000	2.52
air03	10757(10757)	124	340160.00	2.6
air04	8904(8904)	823	56137.00	37.95
air05	7195(7195)	426	26374.00	43.15
arki001	1388(415)	1048	7580902.49	71.46
bell03a	133(39)	123	878430.32	0.26
bell5	104(30)	91	8966406.49	0.81
blend2	353(231)	274	7.6	1.88
cap6000	6000(6000)	2176	2451293.00	0.3
Ex17-Panos-v25	930(480)	561	28.940	9.55
Ex21-Panos-v23	1482(760)	861	19.78	3.94
Luke101min	349(110)	369	2238.99	1.63
Luke102min	349(110)	369	24737.92	1.59
Luke103min	349(110)	369	23294.77	0.71
Luke104min	349(110)	369	20686.26	0.69
Luke105min	349(110)	369	25315.32	4.74
Luke151min	745(241)	775	12268.54	5.9
Luke152min	745(241)	775	25341.5	23.93
Luke153min	745(241)	775	13798.64	9.95
Luke154min	745(241)	775	22372.57	128.54
Luke155min	745(241)	775	15800.29	30.91
Luke101.dat_M2_U	161(105)	256	106132.1	5.66
Luke102.dat_M2_U	161(105)	256	152739.00	1.29
Luke103.dat_M2_U	161(105)	256	101858.78	4.25
Luke104.dat_M2_U	161(105)	256	70225.28	0.26
Luke105.dat_M2_U	161(105)	256	138019.38	2.44

Contribution of each neighborhood

- Search through 10 neighborhoods can be time consuming and our goal is to examine are all of them really necessary
- The first step is to determine the contribution of each neighborhood
- Some of the neighborhoods (N_3, N_4, N_9 i N_{10}) include randomness and we permed 10 restarts with controlled seed values

$$seed(i) = 100 \cdot i$$

- Average values of time (objective function) are used for making decisions

Results: Comparison of neighborhoods

- We noticed that for some of the examples (5 of them) the solution quality is not the same as for VINS
- Therefore, we have two types of results
- For 21 examples with stable objective function value, we ranked neighborhoods according to the running times
- For the remaining 5 instances, solution quality was the ranking criterion
- Smaller rank value corresponds to the better performing neighborhood
- Relative errors

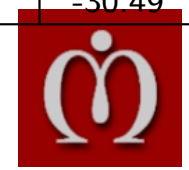
$$f_{dev} = 100 \cdot \frac{f_{SN} - f_{VINS}}{f_{VINS}}, \qquad t_{dev} = 100 \cdot \frac{t_{SN} - t_{VINS}}{t_{VINS}}$$

Comparison results for stable objective function

inst.	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
10teams	7	5	2	9	6	8	4	10	3	1
air03	10	3	4	6	2	1	9	8	5	7
air04	9	5	8	10	4	3	2	1	6	7
air05	1	2	5	10	3	6	9	4	8	7
bell3a	3.5	1.5	9	7	3.5	1.5	5.5	5.5	10	8
bell5	6.5	1.5	10	9	4	1.5	4	4	8	6.5
blend2	8	9	6	5	2	10	1	3	7	4
Ex17-Panos-v25	5	3.5	7	10	2	6	3.5	1	9	8
Ex21-Panos-v23	8	7	1	9	3	10	4	6	5	2
Luke151min	1	4	3	5	6	2	9	10	8	7
Luke153min	1	4	6	8	7	3	2	9	10	5
Luke155min	2	1	5	6	8	3	9	10	4	7
Luke101min	1	9	4	7	3	10	8	2	5	6
Luke102min	1.5	9	5	8	1.5	10	4	3	6	7
Luke104min	7	2	3	4	1	10	8.5	8.5	6	5
Luke105min	10	8	2	5	1	9	7	6	4	3
Luke101.dat_M2_U	4	10	5	8	2	9	3	1	6	7
Luke102.dat_M2_U	9	4	7	5	2	3	10	1	8	6
Luke103.dat_M2_U	1	10	5	7	6	9	2	3	8	4
Luke104.dat_M2_U	8	8	4	6	2	10	1	4	8	4
Luke105.dat_M2_U	3	2	8	4	1	10	5	7	9	6
av.	5.07	5.17	5.19	7.05	**3.33**	6.43	5.26	5.1	6.81	5.6

Relative errors of running times with respect to VINS

inst.	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
10t	122.22	80.95	28.93	919.8	82.54	923.02	78.57	5476.59	30.4	28.81
air03	-34.62	-66.54	-58.35	-51.42	-68.078	-71.92	-45	-46.54	-52.38	-48.85
air04	-51.23	-64.14	-53.54	-49.51	-64.58	-67.54	-76.97	-82.58	-56.56	-54.41
air05	-47.32	-36.66	-26.51	-3.4	-35.75	-20.03	-15.78	-31.39	-18.25	-18.69
bell3a	-61.54	-69.23	-30.77	-45	-61.54	-69.23	-50	-50	-27.31	-43.08
bell5	-86.42	-92.59	-74.44	-79.14	-88.89	-92.59	-88.89	-88.89	-79.51	-87.04
b2	-71.81	-69.68	-81.86	-82.77	-86.7	-59.04	-87.77	-85.64	-77.29	-83.88
Ex17	-85.45	-85.76	-83.5	-74.49	-85.86	-83.77	-85.76	-91.73	-78.95	-79.29
Ex21	40.86	-15.23	-50.86	83.38	-33.5	514.47	-33.25	-17.01	-31.52	-34.67
L151	85.25	3061.36	2993.22	3405.76	3577.29	131.69	3765.08	3992.71	3755.25	3581.53
L153	67.74	1954.57	2014.17	2103.62	2082.51	1837.09	1470.75	2122.61	2181.81	1958.53
L155	-84.7	-89.32	36.53	97.83	216.69	-55.58	221.68	251.02	20.71	191.1
L101	-2.45	200	87.73	106.75	78.53	357.06	151.53	70.55	96.32	97.55
L102	-79.87	-70.44	-77.99	-76.1	-79.87	-20.75	-78.62	-79.25	-77.36	-77.17
L104	2.9	-15.94	-14.78	-14.64	-23.19	14.49	8.7	8.7	-8.7	-14.49
L105	-39.03	-54.64	-75.74	-73.42	-77.22	-50.63	-65.19	-66.03	-73.63	-74.89
L101_U	-37.46	-13.96	-36.4	-31.63	-37.81	-26.15	-37.63	-37.99	-34.98	-33.04
L102_U	-30.23	-34.11	-32.56	-33.33	-36.43	-34.88	17.05	-42.64	-32.33	-32.87
L103_U	-36.71	-6.82	-30.68	-29.18	-30.12	-8.71	-33.65	-32.71	-25.18	-31.29
L104_U	38.46	38.46	34.23	35.38	30.77	69.23	26.92	34.62	38.46	34.23
L105_U	-45.49	-51.23	-23.61	-31.93	-52.05	-8.2	-31.15	-28.69	-20.74	-30.49



Ranks and relative errors for changeable objective function

inst.	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
arki001	9	7	8	3	6	2	10	1	4	5
cap6000	7	3	8	6	9.5	3	9.5	3	3	3
Luke152min	2.5	2.5	7	8	2.5	2.5	10	6	9	5
Luke154min	1.5	8	5	3	8	1.5	8	10	4	6
Luke103min	9.5	2.5	7	5	2.5	2.5	2.5	9.5	6	8
av.	5.9	4.6	7	5	5.7	**2.3**	8	5.9	5.2	5.4

inst.	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
a1	0.0034	0.0032	0.0032	0.0026	0.0031	0.0012	0.0048	-0.0001	0.0028	0.0029
c6	0	0.0021	-0.0010	0.0019	-0.0023	0.0021	-0.0023	0.0021	0.0021	0.0021
L152	0	0	-0.0032	-0.0038	0	0	-0.0092	-0.0026	-0.0042	-0.0016
L154	0	-0.0036	-0.0018	-0.0013	-0.0036	0	-0.0036	-0.0081	-0.0017	-0.0021
L103	-0.0021	0	-0.0016	-0.0003	0	0	0	-0.0021	-0.0004	-0.0021

Tests with subsets of neighborhoods

- None of the neighborhoods performs well on average for the majority of examples
- We want to find a robust subset of neighborhoods
- We combined the four identified neighborhoods in different order
- For all combinations, relative errors with respect to the VINS running times are calculated

Running times relative errors for subset of neighborhoods

inst.	$N_5\,N_1$	$N_1\,N_5$	$N_6\,N_2$	$N_1\,N_6$	$N_1\,N_2\,N_5\,N_6$	$N_5\,N_6\,N_1\,N_2$	$N_2\,N_1\,N_5\,N_6$	$N_2\,N_1\,N_6\,N_5$
10t	17.46	17.46	450.00	418.25	421.03	609.92	442.86	446.83
air03	-28.85	-28.46	-70.38	-68.46	-65.77	-47.69	-62.69	-65.38
air04	-49.14	-49.33	-64.53	-64.58	-64.27	-56.55	-62.40	-62.03
air05	-44.39	-44.28	-20.26	-4.95	-6.03	0.47	-2.49	0.61
a1	-77.05	-81.09	-43.94	-74.29	-70.18	-66.32	-58.05	-57.28
bell3a	-50.00	-61.54	-43.94	-65.38	-65.38	-46.15	-69.23	-50.00
bell5	-74.07	-75.31	-92.59	-91.36	-92.59	-88.89	-92.59	-92.59
b2	-20.74	-41.49	-33.51	-39.36	-36.70	-15.43	-39.89	-46.28
c6	73.33	-36.67	240.00	-40.00	-40.00	246.67	120.00	130.00
Ex17	-53.93	-53.93	-74.35	-52.460	-51.94	-78.64	-53.09	-54.35
Ex21	-12.94	-12.69	298.98	59.90	61.17	364.47	-62.69	-62.94
L151	126.78	78.14	122.71	436.78	825.93	611.53	59.66	59.83
L152	156.41	179.36	-85.17	48.27	-39.20	-37.57	-36.23	-35.73
L153	18.19	6.73	96.38	25.63	71.56	59.80	253.87	263.82
L154	711.37	380.72	502.05	494.59	411.18	511.09	678.91	657.97
L155	-85.86	-84.92	-55.71	-16.11	-80.56	-93.40	-93.14	-89.45
L101	-58.28	-56.44	213.50	40.49	-61.35	-53.99	-31.29	-31.29
L102	-84.28	-81.76	-68.55	-76.10	-83.65	-66.04	-81.76	-82.39
L103	478.87	477.46	50.70	-53.52	483.10	64.79	-39.44	-30.99
L104	-23.19	-21.74	-31.88	34.78	-24.64	-30.43	-57.97	-57.97
L105	-66.88	-65.19	-68.35	-10.55	-55.27	-56.12	-56.33	-56.54
L101_U	-39.58	-32.86	6.01	50.88	54.59	36.04	74.20	75.27
L102_U	-37.21	-44.96	18.60	-10.08	-48.84	27.13	46.51	47.29
L103_U	-0.47	-15.76	18.35	47.76	38.35	46.35	104.24	105.18
L104_U	42.31	50.00	176.92	-19.23	19.23	230.77	203.85	207.69
L105_U	-6.97	-41.80	51.64	81.97	9.02	72.54	88.52	90.16
av.	31.19	**9.99**	56.55	40.64	58.03	82.47	45.13	46.52

Summary and Future work

- The contribution of various neighborhoods and their combinations in VINS are analyzed
- The robust subset of neighborhoods, that performed the best on average for all testes examples, is identified

- We plan to include the parameter settings into the analysis
- Explore some other neighborhoods as well as the order of all neighborhoods (10!=3 628 800 possible permutations)
- Tests should be conducted on other problem instances

Thank you for the attention!

Questions?

Tatjana Davidović
tanjad@mi.sanu.ac.rs