SYM-OP-IS 2015: XLII International Symposium on Operations Research, 2015

The Variable Intensity Neighborhood Search for 0-1 MIP

Petar Jovanović

Institute of Physics Belgrade, Belgrade, Serbia
e-mail: petarj@ipb.ac.rs

Tatjana Davidović

Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia
e-mail: tanjad@mi.sanu.ac.rs

Jasmina Lazić

MathWorks, Inc. Matrix House, Cambridge, United Kingdom
e-mail: jsmnlzc@yahoo.com

Snežana Mitrović Minić

Simon Fraser University, Vancouver, Canada
e-mail: snezanam@sfu.ca

Abstract. We propose a new matheuristic, a heuristic approach based on hybridizing an exact MIP solver with local search through different neighborhoods. This method explores the idea of fixing a subset of variables and invoking exact solver to determine values for remaining variables. The number and variation of variables to be fixed are selected in various ways. Each fixing defines a very large scale neighborhood that is searched by MIP solver. The neighborhood sizes are systematically increased together with the corresponding time limits increasing the intensity of the search. Therefore, we named this method Variable Intensity Neighborhood Search (VINS). The ideas for VINS are drown from two other mathheuristics from the literature: Variable Intensity Local Search (VILS) and Variable Neighborhood Decomposition Search for 0-1 MIP (VNDS-MIP). For the experimental study, we used two types of test examples, benchmarks from the MIPLIB 3.0 and instances of the real life problem of routing barge container ships. To examine the performance of the proposed approach, we compare it against VNDS-MIP. Our preliminary experimental results show that on average our VINS is producing high quality solutions very fast, outperforming the other method with respect to solution quality or running time, and sometimes both.

Keywords: Mixed-Integer Programming, Local Search, Combinatorial Optimization, Hybrid Heuristics.

1. Introduction

Discrete optimization problems are very common in real-world applications. However, the size of instances usually exceeds the capacity of computing resources, despite today's very rapid technological development. Therefore, new and efficient solution methods are always necessary.

Hybridization became a very popular approach to increasing the performance of metaheuristic methods [1, 2]. Special type of hybrids, involving combination of an exact solver and a metaheuristic method are explored in the last decade [3–7]. These hybrids are known under the common name *model–based metaheuristics* or *matheuristics* [8].

Inspired by the general purpose matheuristic Variable Neighborhood Decomposition Search for 0-1 MIP (VNDS-MIP) proposed in [5] and the Variable Intensity Neighborhood Search (VINS), a specialized hybrid heuristic for multi-resource generalized assignment problem proposed in [6] and later described in detail in Chap. 10 of [8], we develop a new method that combines and improves good characteristics of its ancestors. To evaluate the performance of our VINS, we compare it against VNDS-MIP on two types of test examples, benchmarks from the MIPLIB 3.0 from (http://miplib.zib.de/miplib3/miplib.html), and barge container ships routing instances [7]. Our preliminary experimental results show that, for most of the examples, our VINS is superior to VNDS-MIP with respect to both solution quality and running time.

The rest of this paper is organized as follows. In Sect. 2 we recall the Mixed Integer Programming (MIP) formulation for optimization problems. Sect. 3 is devoted to the proposed VINS approach. Experimental study is given in Sect. 4, while Sect. 5 contains concluding remarks and directions for future work.

2. Problem description

A linear Mixed Integer Program (MIP) is a model of an optimization problem whose set of variables can be divided into two nonempty subsets: subset of integer and/or binary variables and a subset of continuous variables. The constraints are linear inequalities, and the objective function is also linear. Thus, MIP can be written as

$$\min \ f(x) = c^T \cdot x \tag{1}$$

$$A^T \cdot x \leqslant b, \tag{2}$$

where matrix A and vectors c and b represent problem parameters. Vector $x = (x_1, \ldots, x_n)$ contains the decision variables (unknowns) whose values need to be determined in such a way that $f(x)$ is minimized and all the constrains are satisfied. The variables x_i, $i = 1, \ldots, n$ can take binary, integer or continuous values, i.e., $x \in \mathcal{B} \times \mathcal{Z} \times \mathcal{R}$, where $\mathcal{B} = \{x_k, x_k \in \{0, 1\}\}$, $\mathcal{Z} = \{x_k, x_k \in \mathbb{Z} \setminus \{0, 1\}\}$, and $\mathcal{R} = \{x_k, x_k \in \mathbb{R} \setminus \mathbb{Z}\}$.

Many real life problems may be represented in a MIP form. Several well known examples include vehicle routing problems, scheduling, packing, facility location, network flow. There is variety of efficient approaches for solving these problems either exactly or heuristically. However, real life instances are usually of very large sizes and the development of new powerful heuristics is often needed.

3. VINS - The proposed matheuristic

VINS is a generalization of VNDS-MIP matheuristic [5] which defines new patterns for subproblem generation by fixing and relaxing subsets of problem variables. First step of the algorithm is to solve the linear relaxation of MIP. After that it finds the first feasible solution for MIP. The distances between each corresponding variable value in the solution of linear relaxation and the first feasible solution are calculated, and the variables are sorted by these distance values. Variables which are close to the value in linear relaxation solution are considered to offer less space for improvement.

As in VNDS-MIP, neighborhoods are defined in the space of variable states. A variable state is said to be fixed if the optimizer is not allowed to change the value of that variable during the optimization process. Otherwise, a variable state is relaxed, meaning that the optimizer can freely adjust the value of the variable. The neighborhoods have the following patterns for each iteration of the search:

- N1: $\alpha\%$ 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 $\alpha\%$ variables are released;
- N4: at 10 random positions, $\alpha/10\%$ variables are released;
- N5: $\alpha\%$ of the best variables are released;
- 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.

Neighborhood N1 is in line with the original VNDS-MIP method described in [5] where the parameter α was changing from 10 to 100 by an increment of 10. Terms "best" and "worst" variables here refer to lowest and highest distance of the variable value from the value in linear relaxation solution, respectively. Parameter α defines the percentage of variables to be released and used in the search, i.e. the neighborhood size. The neighborhood change patterns are iterated in round robin fashion. When they are all explored, the neighborhood size is increased, and the search is restarted in N1. Whenever an improvement is achieved, the new best solution is added as a constraint on the solution value, and the variables are resorted according to the distance between values in current solution and the linear relaxation solution. The details of the proposed procedure are given as Algorithm 1.

4. Experimental results

The algorithm is coded in C++ and executed on a 3.9 GHz quad core Intel i7 4770 CPU with 8 GB of RAM. As the exact MIP solver CPLEX 11.2 is used. To assure fair comparison, VNDS-MIP is executed in the same environment. The parameter α was taking values 40%, 60%, 80% and 100%, and the time limit for each neighborhood size was set respectively to 200, 400, 1000 and 2100 seconds, giving 1 hour total execution limit per problem instance.

Table 1 contains the comparison results on barge container ships routing problem instances. In the first three columns the problem size (expressed by number of ports n, total number of variables with number of binary variables in brackets, and the number of constraints, respectively) is given. The remaining two

Algorithm 1 VINS

Input: MIP, $alphas$, $time\_limits$
  Solve linear relaxation of MIP to obtain $linx$.
  Calculate x, the first feasible solution for MIP.
  $bestx \leftarrow x$.
  $improvement \leftarrow true$.
  while $t < Tlim$ **do**
    if $improvement$ **then**
      Sort variables according to $x - linx$.
      $improvement \leftarrow false$.
    end if
    $N \leftarrow NextNeighborhood$
    if $N = null$ **then**
      $\alpha \leftarrow NextAlpha$
      if $alpha = null$ **then**
        break
      end if
      $N \leftarrow N1$
    end if
    $Release(N, \alpha)$.
    $x \leftarrow MIPSolve(MIP, time\_limit)$.
    $improvement \leftarrow improved(x)$
    if $improvement$ **then**
      $bestx \leftarrow x$.
      Add objective constraint $f(x) < f(bestx)$.
    end if
  end while

column pairs contain the best objective function value and the corresponding running time for VNDS-MIP and VINS. The values in each cell represent average results for 5 instances of the same size. The results presented in Table 1 show that solution quality is the same for both methods on smaller examples. However, VINS is superior with respect to running time. For the largest instances, as well as in total, VINS outperformed VNDS-MIP with respect to both solution quality and running time.

The comparison results on miplib3 instances are presented in Table 2. The instance name is given in the first column of this table. The next two columns contain the size of the instance: total number of variables and number of binary variables in brackets, and the number of constraints, respectively. The next two column pairs contain the best objective function value and the corresponding running time for VNDS-MIP and VINS. The last two columns show ranks of VNDS-MIP and VINS, respectively. Rank of a method is defined as 1 if it produces better solution and 2 otherwise. If the solution quality is the same, both methods have rank equal to 1.5. As can be seen from Table 2, VINS performs slightly better then VNDS-MIP on average.

Table 1. Computational results on barge container ships routing problem instances

	number of		VNDS-MIP		VINS	
n	var.	constr.	best.obj.	min.time	best.obj.	min.time
10	352 (110)	398	**-23274.62**	12.65	**-23274.62**	**3.36**
15	752 (240)	818	**-17916.31**	131.15	**-17916.31**	**78.75**
20	1302 (420)	1388	**-25016.45**	3875.02	-25000.51	**861.33**
25	2002 (650)	2108	-25247.86	3727.64	**-26663.73**	**1509.67**
av.			-22863.82	1936.61	**-23213.80**	**613.28**

5. Conclusion

A new matheuristic VINS for solving large scale MIPs is designed combining two other approaches VILS and VNDS-MIP and tested on two types of test examples. VINS is a general purpose hybrid method that searches through several neighborhoods differing with respect to both type and size. Our preliminary experimental study shows that, in most of the examples, VINS is outperforming previous method producing high quality solutions in reasonable time.

The further development of VINS should include parameter tuning, the evaluation of neighborhoods efficiency, their proper selection and ordering, and learning how to order the decision variables and reduce the search space relying on the availability of previously generated solutions.

Acknowledgements. *This work has been partially supported by Serbian Ministry of Science, grant Nos. ON174010, ON174033 and ON171017 and NSERC.*

References

[1] E.-G. Talbi. A taxonomy of hybrid metaheuristics. *J. Heur.*, 8:541–564, 2002.

[2] E.-G. Talbi, ed. *Hybrid metaheuristics*. Springer, 2013.

[3] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(2):23–47, 2003.

[4] P. Hansen, N. Mladenović, and D. Urošević. Variable neighbourhood search and local branching. *Comput. Oper. Res.*, 33(10):3034–3045, 2006.

[5] J. Lazić, S. Hanafi, N. Mladenović, and D. Urošević. Variable neighbourhood decomposition search for 0–1 mixed integer programs. *Computers and Operations Research*, 37(6):1055–1067, 2010.

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

[7] V. Maraš, J. Lazić, T. Davidović, and N. Mladenović. Routing of barge container ships using MIP heuristics. *Applied Soft Computing*, 13(8):3515–3528, 2013.

[8] V. Maniezzo, T. Stützle, and S. Voss, editors. *Matheuristics: hybridizing metaheuristics and mathematical programming*, volume 10. Springer, 2009.

Table 2. Computational results on miplib3 instances

			VNDS-MIP		VINS		Ranks	
inst.	no.var.	no.constr.	best.obj.	min.time	best.obj.	min.time	VNDS-MIP	VINS
10teams	2025 (1800)	230	**924**	25.899	928	6.578	1	2
air03	10757 (10757)	124	340160	2.609	340160	0.703	1.5	1.5
air04	8904 (8904)	823	56137	73.203	56137	12.078	1.5	1.5
air05	7195 (7195)	426	26374	129.516	26374	16.265	1.5	1.5
arki001	1388 (415)	1048	**7580814.512**	297.031	7589104.642	13.75	1	2
bell03a	133 (39)	123	878430.316	4.922	878430.316	1.969	1.5	1.5
bell5	104 (30)	91	8966406.492	0.469	8966406.492	0.156	1.5	1.5
blend2	353 (231)	274	8.077526	1.203	**7.598985**	0.047	2	1
cap6000	6000 (6000)	2176	**-2451377**	57.313	-2451239	0.219	1	2
dano3mip	13873 (552)	3202	696.6666667	1932.344	**691.1352941**	3341.61	2	1
danoint	521 (56)	664	66.5	1.469	**65.66666667**	1.203	2	1
dcmulti	548 (75)	290	188194.6	3599.969	**188182**	0.109	2	1
dsbmip	1937 (160)	1182	-305.198175	1.406	-305.198175	0.187	1.5	1.5
egout	141 (55)	98	568.1007	0.078	568.1007	0.094	1.5	1.5
enigma	100 (100)	21	0	0.078	0	0	1.5	1.5
fast0507	63009 (63009)	507	174	96.641	174	237.812	1.5	1.5
fiber	1298 (1254)	363	405935.18	2.313	405935.18	0.188	1.5	1.5
fixnet6	878 (378)	478	3983	0.75	3983	1.453	1.5	1.5
gen	870 (144)	780	112313.3627	0.094	112313.3627	0.187	1.5	1.5
gesa2	1224 (240)	1392	25779856.37	6.969	25779856.37	1.954	1.5	1.5
gesa2_o	1224 (384)	1248	**25779856.35**	32.625	25780031.43	5.235	1	2
gesa3	1152 (216)	1368	27991042.65	15.922	27991042.65	2.953	1.5	1.5
gesa3_o	1152 (336)	1224	27991042.65	4.828	27991042.65	0.219	1.5	1.5
gt2	188 (24)	29	21166	0.031	21166	0	1.5	1.5
harp2	2993 (2993)	112	-73868341	3183.375	**-73899770**	30.062	2	1
khb05250	1350 (24)	101	106940226	3.672	106940226	0.484	1.5	1.5
l152lav	1989 (1989)	97	4722	6.641	4722	0.828	1.5	1.5
lseu	89 (89)	28	1120	0.703	1120	0.313	1.5	1.5
markshare1	62 (50)	6	5	911.025	5	2395.391	1.5	1.5
markshare2	74 (60)	7	**14**	1649.634	15	2576.063	1	2
mas74	151 (150)	13	11801.18573	1135.188	11801.18573	215.282	1.5	1.5
mas76	151 (150)	12	40005.05414	0.859	40005.05414	7.078	1.5	1.5
misc03	160 (159)	96	3360	0.328	3360	0.578	1.5	1.5
misc06	1808 (112)	820	12850.86074	0.828	12850.86074	0.437	1.5	1.5
misc07	260 (259)	212	2810	3.531	2810	21.828	1.5	1.5
mitre	10724 (10724)	2054	115155	2.845	115155	0.515	1.5	1.5
mkc	5325 (5323)	3411	-558.544	1590.234	**-563.846**	202.031	2	1
mod008	319 (319)	6	307	0	307	0.031	1.5	1.5
mod010	2655 (2655)	146	6548	0.359	6548	0.25	1.5	1.5
mod011	10975 (96)	4480	-54558535.01	384.321	-54558535.01	19.328	1.5	1.5
noswot	128 (75)	182	-41	0.016	-41	1.703	1.5	1.5
nw04	87482 (87482)	36	16862	32.902	16862	7.453	1.5	1.5
p0033	33 (33)	16	3089	0.047	3089	0.125	1.5	1.5
p0201	201 (201)	133	7615	0.75	7615	0.203	1.5	1.5
p0282	282 (282)	241	258411	0.703	258411	0.093	1.5	1.5
p0548	548 (548)	176	8691	0.172	8691	0.328	1.5	1.5
p2756	2756 (2756)	755	3124	6.422	3124	0.281	1.5	1.5
pk1	86 (55)	45	11	45.484	**10.99999995**	29.406	2	1
pp08a	240 (64)	136	7350	4.641	7350	0.344	1.5	1.5
pp08aCUTS	240 (64)	246	7350	3.016	7350	0.422	1.5	1.5
qiu	840 (48)	1192	-132.8731369	2111.469	-132.8731369	6.203	1.5	1.5
qnet1	1541 (1288)	503	16029.69268	2.938	16029.69268	3.36	1.5	1.5
qnet1_o	1541 (1288)	456	16029.69268	3.125	16029.69268	0.485	1.5	1.5
rentacar	9559 (55)	6803	30356760.98	3.125	30356760.98	0	1.5	1.5
rgn	180 (100)	24	82.19999924	0.031	82.19999924	0.031	1.5	1.5
rout	556 (300)	291	1077.56	117.844	1077.56	22.641	1.5	1.5
set1ch	712 (240)	492	54537.75	8.109	54537.75	1.203	1.5	1.5
seymour	1372 (1372)	4944	425	89.078	**423**	2556.313	2	1
stein27	27 (27)	118	18	0	18	0.046	1.5	1.5
stein45	45 (45)	331	30	0.078	30	0.328	1.5	1.5
swath	6805 (6724)	884	**471.033123**	1326.672	478.027556	428.827	1	2
vpm1	378 (168)	234	20	0.172	20	0	1.5	1.5
vpm2	378 (168)	234	14	0.109	**13.75**	0.125	2	1
AVERAGE			2113361.622	300.287746	**2112999.07**	**193.26**	1.52	**1.48**