Chapter 10
Variable Intensity Local Search

Snežana Mitrović-Minić and Abraham P. Punnen

Abstract This chapter considers a local search based heuristic framework for solving the mixed-integer programming problem (MIP) where a general purpose MIP solver is employed to search the associated neighborhoods. The associated neighborhood search problems are MIPs of smaller sizes. The neighborhoods are explored in varying the intensity by changing time and size parameters. This local search can be viewed as a combination of very large scale neighborhood (VLSN) search and variable neighborhood search (VNS). The approach has been implemented to solve two integer programming problems: the generalized assignment problem, and the multi-resource generalized assignment problem. Encouraging computational results have been achieved.

10.1 Introduction

In this chapter we consider a local search algorithm for the mixed-integer programming problem (MIP) based on the well known k-exchange neighborhood. Unlike traditional k-exchange based local search that considers small values of k, we use large values k. An MIP solver is used to explore the neighborhoods for improved solutions. The neighborhoods size and search intensity are controlled by two search-intensity parameters. Our algorithm in many cases does not explore the k-exchange neighborhood optimally but performs only an approximate search. Thus, we are exploring only a partial k-exchange neighborhood, for various values of k.

Several local search algorithms from the optimization literature that partially explore k-exchange neighborhoods—and are classified as variable depth methods—include the Lin-Kernighan algorithm for TSP [11] and ejection

Snežana Mitrović-Minić · Abraham P. Punnen
Department of Mathematics, Simon Fraser University, Surrey, Canada
e-mail: {snezanam,apunnen}@sfu.ca

V. Maniezzo et al., (eds.), *Matheuristics*, Annals of Information Systems 10,
DOI 10.1007/978-1-4419-1306-7_10,

chain algorithms for various combinatorial optimization problems [10]. Although our algorithm is a local search, considering its linkages with VLSN search [1, 2] and VNS [16], we call it *variable intensity local search* (VILS).

Using an MIP solver within local search to explore neighborhoods received considerable attention in the recent years: [5, 6] for the general MIP, [3, 22, 21, 23] for variations of the vehicle routing problems, [18, 15, 14] for the variations of the generalized assignment problem. The algorithm discussed in this chapter is a generalization of the heuristics developed for the generalized assignment problem (GAP) [15] and the multi-resource generalized assignment problem (MRGAP) [14].

This chapter is organized as follows. In Section 10.2 we introduce the general VILS framework. Section 10.3 gives brief description of our experimental studies on GAP and MRGAP problems whose details are reported in [15, 14]. Concluding remarks are given in Section 10.4.

10.2 The General VILS Framework

The VILS algorithm is a local search algorithm for MIP using the k-exchange neighborhood for varying values of k, adjusted systematically during the algorithm. The resulting neighborhoods are searched approximately using an MIP-solver with varying intensity level. Consider the MIP

$$\begin{array}{lll} \text{MIP:} & \text{Maximize} & CX \\ & \text{Subject to} & AX = b \\ & & X \geq 0, \quad X \text{ integer,} \end{array}$$

where $X^T = (x_1, x_2, \ldots, x_n)$ is a vector of n variables, and the problem parameters are: $A = (a_{ij})$ which is an $m \times n$ matrix, $b^T = (b_1, b_2, \ldots, b_m)$ which is an m-vector, and $C = (c_1, c_2, \ldots, c_n)$ which is an n-vector. For simplicity of presentation, we avoid real variables (which are never set to a fixed value during the course of the algorithm) in the above description of an MIP. Let $\hat{X}$ be a feasible solution to the MIP. A *binding set* $\hat{S}$ is a subset of variable indices $\{1, 2, \ldots, n\}$ which defines a k-exchange neighborhood. The neighborhood $N(\hat{X})$ consists of all solutions of the MIP whose j^{th} variable is equal to the value of the j^{th} variable in $\hat{X}$ for all $j \in \hat{S}$, *i.e.*

$$N(\hat{X}) = \{X \mid x_j = \hat{x}_j, \ \forall j \in \hat{S} \text{ and } X \text{ is a feasible solution to the MIP}\}$$

The neighborhood $N(\hat{X})$ can be searched for an improving solution by solving the following restricted MIP

$$\begin{array}{lll} \text{MIP}(\hat{S}): & \text{Maximize} & \displaystyle\sum_{j \in N \setminus \hat{S}} c_j x_j \\ & \text{Subject to} & \end{array}$$

$$\sum_{j \in N \setminus \hat{S}} a_{ij} x_{ij} = \hat{b}_i, \text{ for } i = 1, 2, \ldots m$$
$$x_j \geq 0, \quad x_j \text{ integer for } j \in N \setminus \hat{S}$$

where $\hat{b}_i = b_i - \sum_{j \in \hat{S}} a_{ij} \hat{x}_{ij}$. If $\bar{X} = \{\bar{x}_j \ : \ j \in N \setminus \hat{S}\}$ is a feasible solution to MIP($\hat{S}$) then the n-vector X defined by

$$x_j = \begin{cases} \hat{x}_j, & \text{if } j \in \hat{S} \\ \bar{x}_j, & \text{otherwise} \end{cases}$$

is a feasible solution to MIP and $X \in N(\hat{X})$. We call such a solution X *the solution augmented by* $\bar{X}$.

The complexity of MIP($\hat{S}$) in practice depends primarily on the size of $\hat{S}$ although other factors are also involved. If $|\hat{S}|$ is large (and hence $|N \setminus \hat{S}|$ is small) MIP($\hat{S}$) can normally be solved optimally in reasonable time using an MIP solver. However, in this case, $|N(\hat{X})|$ is likely to be small, limiting the power of the local search. If $|\hat{S}|$ is small, then $|N(\hat{X})|$ is likely to be large yielding a more powerful search but the time for searching the neighborhood using the MIP solver could be large. Thus the efficiency of the local search using the MIP solver depends on our ability to guide the search appropriately by controlling the size of $\hat{S}$, the time invested in searching $N(\hat{X})$, and the choice of elements in $\hat{S}$.

In the VILS algorithm, we keep six major parameters: p is the cardinality of the binding set $\hat{S}$, p_0 is its initial value, Δp is the downward step size to decrease the value of p; t is the time limit for the MIP solver, t_0 is its initial value, and Δt is the upward step size of t.

Initially, we set a large value of p yielding smaller neighborhoods. The search times for these neighborhoods are set to small values and the local search is carried out until a decision is made to intensify the search. At this stage, the value of p is decreased (and thereby increasing the neighborhood size) and the time limit for searching the neighborhood is increased. This process is continued until a prescribed stopping criterion is reached.

The control mechanism using the time limits and the systematic intensification of the search resulted in good experimental results. Different selections of the binding sets $\hat{S}$ yield different neighborhoods, and they are normally problem specific. Assume that L is the number of different neighborhoods. If no improvement is obtained after employing several binding set selection rules, the search intensity is increased by decreasing p and increasing t. A high level description of the VILS algorithm is given in Figure 10.1.

The neighborhoods and intensity-search schemata are problem specific. Neighborhoods may be designed using any existing or new strategy for choosing a binding set. Our strategies for choosing binding sets may be summarized as follows. A criterion for "good" variables is chosen beforehand, and variables are fixed in order of "goodness". In the iteration where p variables

```
The VILS Algorithm
Input: Problem instance P;
    the stopping criterion and the intensity-search change criterion;
    p_0, Δp; t_0, Δt
begin
    generate feasible solution X̂
    i ⇐ 0
    p ⇐ p_0
    t ⇐ t_0
    while (the stopping criterion is not satisfied) do
        choose the binding set S_i such that |S_i| = p and
        generate the neighborhood N_i
        /* search the neighborhood */
        Solve the problem MIP(S_i) by running the MIP solver for time t
        Let X̄ be the best solution obtained
        Compute the augmented solution X'
        /* update the current solutions */
        if (CX' < CX̂) then
            X̂ ⇐ X'
        end if
        i ⇐ (i + 1) mod L
        if (the intensity-search change criterion is satisfied) then
            p ⇐ p − Δp
            t ⇐ t + Δt
        end if
    end while
    return X̂
end
```

Fig. 10.1 Outline of the VILS Algorithm.

have to be fixed, the following are the neighborhoods for the GAP and the MRGAP with m machines.

1. For each machine, fix p/m "best" variables out of the value-one variables.
2. For each machine, fix p/m "worst" of the value-one variables.
3. For each machine, fix $p/(m/2)$ "best" and $p/(m/2)$ "worst" of the value-one variables.
4. For half of the machines, fix p/m "best", and for the other half of the machines, fix p/m "worst" of the value-one variables.
5. Fix "best" p of the value-one variables.
6. Fix "worst" p of the value-one variables.
7. Fix $p/2$ "best" and $p/2$ "worst" of the value-one variables.
8. Controlled random fixing: fix $p/10$ random variables in the "best" 10% of the value-one variables, fix $p/10$ random variables in the next "best" 10% (11% to 20%) of the value-one variables, etc.
9. Meta-neighborhood: fix certain sequences of the value-one variables in the given "goodness" order.

The variable "goodness" criteria depends only on the initial problem parameters, and thus the variables can be ordered by their goodness in a preprocessing step. An example of a "goodness" criterion we used for the GAP is: A "good" variable is one with smaller ratio cost per resource needed. Further details about the neighborhoods used in our two experimental studies may be found in [14, 15].

Since the truncation of the current solution $\hat{X}$ is a feasible solution to MIP(S_i), we supply it to the MIP solver. To test the efficiency of the algorithm we considered two specific problems: the GAP which is well studied in literature [4, 12, 13, 19, 24, 25] and its generalization the MR-GAP [7, 8, 9, 17, 20]. Our experimental studies, algorithm parameters, and results are summarized in the next section.

10.3 Experimental Studies

We have implemented the VILS for the GAP and MRGAP in C++ and tested on a Dell workstation with one Intel Xeon 2.0GHz processor, 512 MB memory, GNU g++ compiler version 3.2, and Linux (Mandrake 9.2) operating system. To search the neighborhoods we have used CPLEX 9.1 with Concert Technology.

The stopping criterion has been taken according to the time limits used in [24, 26, 25]. The intensity-search change criterion has been: "solution has not improved in 3 iterations" although we also experimented with values 2, 4 and 5. Preliminary studies have also shown that an appropriate number of different neighborhoods (the binding strategies) L should be between 4 and 10, when the intensity-search change criterion is 2 or 3 to assure that each neighborhood type is searched once in every two or three intensity settings.

We have experimented with different intensity schedules with variety of combinations of changing time limits and binding set size alternatively and simultaneously. However, more complicated schedules, as well as more granular schemas, have not shown any additional advantages. When a number of iterations does not generate an improving solution, the simple increase in time limit and neighborhood size almost always produces improved solution. Further research towards reactive VILS is in progress, where initial neighborhood size and time limit as well as the steps would be chosen automatically.

For the GAP, standard benchmark large instances of types C, D, and E, with 900 and 1600 jobs, generated by [24] were used as the test bed. Nine out of eighteen solutions achieved by VILS were equal or better in quality compared to the solutions when tabu search by [25] was run only once. (Six solutions were better.) When tabu search was run five times [25], it achieved better results for all but two instance. In other five instances the solutions were the same.

For the MRGAP, our testbed consists of MRGAP instances generated by [26] from the standard benchmark GAP instances of types C, D and E with 100 and 200 tasks (which were generated by J.E. Beasley). We have tested the VILS with two different intensification schemes: $Sch1$ and $Sch2$ (details may be found in [15, 14]). The solutions achieved by VILS are better or equal in quality compared to the solutions reported in the literature, with a few exceptions when tabu search by [26] or CPLEX achieved better solutions.

For the D instances, the best solutions were achieved by VILS with intensification schedule $Sch1$ in 7 cases, by VILS with intensification schedule $Sch2$ in 10 cases, and by CPLEX in 10 cases. Unique best solutions were achieved by VILS ($Sch1$), VILS ($Sch1$), and CPLEX in 6, 8, and 7 instances, respectively. For the E instances, the best solutions were achieved by VILS ($Sch1$), VILS ($Sch1$), tabu search [26], and CPLEX in 16, 14, 8, and 12 instances, respectively. Unique best solutions were achieved by VILS ($Sch1$), VILS ($Sch1$), tabu search [26], and CPLEX in 6, 3, 2, and 1 instances, respectively.

10.4 Conclusion

In this chapter we proposed an implementation of a local search framework, called Variable Intensity Local Search, for solving mixed-integer programming problems. The neighborhoods are explored using a general purpose MIP solver. Depending on the binding sets, the neighborhoods could be of different structure and hence the algorithm can be viewed as a variable neighborhood search. In addition, since some of the search neighborhoods could be very large, the algorithm can be viewed as a very large scale neighborhood search as well. We have done two experimental studies solving GAP and MRGAP which showed that good quality solutions can be reached in a reasonable time. We are in the process of conducting an experimental study on a facility location problem and on a general 0-1 MIP.

The major advantage of the approach is its local search framework simplicity, and ability to achieve satisfactory results by controlling the intensity and depth of the neighborhood search. Furthermore, our heuristic can be embedded in any metaheuristic.

Acknowledgements This work is partially supported by an NSERC discovery grant awarded to Abraham P. Punnen.

References

1. R.K. Ahuja, O. Ergun, and A. Punnen. A survey of very large scale neighborhood search techniques. *Discrete Applied Mathematics*, 23:75–102, 2002.
2. R.K. Ahuja, O. Ergun, and A. Punnen. Very large scale neighborhood search: Theory, algorithms, and applications. In T. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, volume 10 of *Computer and Information Science Series*. Chapmann and Hall, CRC Press, 2007.
3. R. Bent and P. V. Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33:875–893, 2006.
4. D. Cattrysse and L.N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60:260–272, 1992.
5. E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102:71–90, 2005.
6. M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003.
7. B. Gavish and H. Pirkul. Allocation of databases and processors in a distributed computing system. In J. Akoka, editor, *Management of Distributed Data Processing*. North-Holland Publishing Company, Amsterdam, 1982.
8. B. Gavish and H. Pirkul. Computer and database location in distributed computer systems. *IEEE Transactions in Computing*, 35:583–590, 1986.
9. B. Gavish and H. Pirkul. Algorithms for the multi-resource generalized assignment problem. *Management Science*, 37:695–713, 1991.
10. F. Glover. New ejection chain and alternating path methods for traveling salesman problem. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Development in Their Interfaces*, pages 491–507. Pergamon, Oxford, 1992.
11. S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.
12. H.R. Lourenço and D. Serra. Adaptive approach heuristic for the generalized assignment problem. Technical report, Department of Economics and Management, Universitat Pompeu Fabra, R. Trias Fargas 25-27, 08005 Barcelona, Spain, 1998.
13. S. Martello and P. Toth. An algorithm for the generalized assignment problem. In J.P. Brans, editor, *Operational Research'81*, pages 589–603. North-Holland, 1981.
14. S. Mitrovic-Minic and A.P. Punnen. Local search intensified: Very large-scale variable neighborhood search for the multi-resource generalized assignment problem. Submitted for publication.
15. S. Mitrovic-Minic and A.P. Punnen. Very large-scale variable neighborhood search for the generalized assignment problem. *Journal of Interdisciplinary Mathematics*, 11(5):653–670, 2008.
16. N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.
17. R.A. Murphy. A private fleet model with multi-stop backhaul. Working paper 103, Optimal Decision Systems, Green Bay, WI54306, 1986.
18. T. Oncan, S.N. Kabadi, K.P.N. Nair, and A.P. Punnen. VLSN search algorithms for partitioning problems using matching neighbourhoods. *The Journal of the Operational Research Society*, 59:388–398, 2008.
19. I.H. Osman. Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches. *OR Spektrum*, 17:211–225, 1995.
20. H. Pirkul. An integer programming model for allocation of databases in a distributed computer system. *European Journal of Operational Research*, 26:401–411, 1986.
21. D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

22. S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.
23. G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.
24. M. Yagiura, T. Ibaraki, and F. Glover. An ejection chain approach for the generalized assignment problem. *INFORMS Journal on Computing*, 16:133–151, 2004.
25. M. Yagiura, T. Ibaraki, and F. Glover. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, 169:548–569, 2006.
26. M. Yagiura, S. Iwasaki, T. Ibaraki, and F. Glover. A very large-scale neighborhood search algorithm for the multi-resource generalized assignment problem. *Discrete Optimization*, 1(1):87–98, 2004.