Variable Neighbourhood Heuristic for the Resource Levelling Problem

Allen R. Mushi
Department of Mathematics, University of Dar es *salaam*
Box 35062 Dar es salaam, Tanzania
amushi@maths.udsm.ac.tz/allen.mushi@gmail.com

***Abstract*—This paper explores the use of a heuristic method based on Simulated Annealing with variable neighbourhood as an optimisation technique for the Resource Levelling Problem. Resource levelling is one of the resource constrained project scheduling problems. Its objective is to adjust the start dates of individual activities within a project in order to minimise the maximum resource level such as manpower, over the duration of the project while maintaining all given precedence relations. It is an NP-Hard combinatorial optimisation problem with major applications in manufacturing industries. It is concluded that, Increasing Variable Neighbourhood on Simulated Annealing is a reasonably good heuristic for the Resource Levelling Problem**

Keywords-Resource Levelling, Simulated Annealing, Combinatorial Optimization, Variable Neighbourhood

I. INTRODUCTION

Resource levelling Problem (RLP) deals with one of the most crucial aspects of project planning. Usually scheduling a project consists of determining a set of starting times for the activities of the project in such a way that the precedence constraints between them are satisfied and the total completion time is minimised. If there are no resource constraints, problems of realistic size can be solved by the critical path and network flow techniques [1]. However critical path methods assume unlimited availability of resources which is unrealistic. Often resources are scarce and their utilization needs to be optimized. The activity start times developed from critical path techniques imply specified patterns or 'profiles' of resource usage over time. When resource availability levels are checked against the required levels of demand, the problem of resource allocation arises. It may happen that demands exceeds availability levels in certain time periods or that the variation in resource profiles is not acceptable, and there is a need to reduce excessive peaks and 'smooth' the profiles of usage.

The objective of the resource levelling process is to 'smooth' as much as possible the profiles of usage over time, within the given project duration. This is accomplished by judicious rescheduling of activities within their available slack (floating time) to give the most acceptable profile.

Clearly, there is an exponential number of combinations of starting times (feasible solutions) to be considered in the search for the optimal profile. This is an NP-Hard combinatorial optimisation problem and thus no exact algorithm is currently known for its solution [2].

Many solution approaches have been attempted over the years. Exact solution techniques include the works on mathematical programming models by Stevens [3], Martinez and Ioannou [4], and Son and Skibniewski [5]. However, these methods have not been successful in solving real-life size problems. An extensive literature is available that shows many attempts in the development of heuristics for resource levelling problem [6]-[10]. The early and famous heuristics includes the work by Burgess and Killebrew [11], which is a very simple method. Development of better heuristics which seeks to explore the possibilities of jumping out of local minima and converging into global optimal solution has been developed over the years. Recent heuristics includes the use of Petri Nets [12], Genetic Algorithms based [13],[14],[15], Ant Colony Optimization [16], Particle Swam Intelligence [17], Dynamic Programming [18], and tree-based methods [19]. Some commercial software packages are also available with varying performances. Kastor and Sirakoulis [20] presented an evaluation of several packages and highlighted some advantages and drawbacks inherent in each of the evaluated packages.

Simulated Annealing is one of the most successful global heuristic techniques and has been implemented in many combinatorial optimization problems. Son and Skibniewski [5] developed a multi-heuristic model called hybrid model combining the local optimizer with simulated annealing for resource leveling. However, most of these heuristics were developed for specific problems especially in construction industry.

There is a need and ample scope for the development of new heuristics for effective scheduling and levelling. This paper focuses on the application of Simulated Annealing with variable neighbourhood. Most of the application of Simulated Annealing employs a fixed neighbourhood structure. The structure of the Resource Levelling Problem shows that a variable neighbourhood may improve the performance on the search space. Furthermore, the application of the algorithm is based on randomly generated problems which include general features that may apply to many possible applications.

The following sections are organised as follows; section II briefly introduces the formulation of Resource Levelling

Problem, followed by presentation of the variable neighbourhood Simulated Annealing in section III. Generation of a random set of problems is discussed in section IV, followed by experimental results in section V. Finally, section VI gives a conclusion and further research directions.

II. Mathematical Programming Formulation of RLP

There are many possible formulations of the problem, each with different qualities. Mushi and O'heigeartaigh [21] discuss various formulations of the problem including re-formulations of some of the constraints. The following is one of the proposed formulations;

Let x_i Start time of activity *i*.

- r_i = Resource level corresponding to activity *i*. (the amount of work units required to be performed on an activity per unit time)
- p_i = Processing time of activity *i*.
- $S_i = [x_i, x_i + p_i - 1]$; Set of points covered by activity *i* on the planning horizon, given that it starts at x_i.
- N = {1,2,...,T}; set of all points on the planning horizon
- h_t = Total resource level in the resource profile at time t (Height of the resource profile at time t) i.e. $h_t = \sum_{i \ni t \in S_i} r_i$.
- n = Total number of activities
- T = The length of planning horizon (Project duration)=|*N*|
- H = Set of all arcs (*i,j*) in the network such that i precedes *j*.

Then the RLP is as follows;

$$Minimize(\max .\,_{\forall t \in N}\{h_t\})$$

$$subject\ to.\ x_j - x_i \geq p_i \quad \forall (i,j) \in H$$

$$x \in Z_+^n$$

Or

$$Minimize\ \ w$$

$$subject\ to.\ h_t \leq w \quad \forall t \in N \qquad \text{(I)}$$

$$x_j - x_i \geq p_i \quad \forall (i,j) \in H \qquad \text{(II)}$$

$$x \in Z_+^n,$$

$$w \in Z_+$$

$$h \in Z_+^T$$

Constraint (I) ensures that the min-max objective function is satisfied, while constraint (II) ensures that precedence constraints are respected during movement of start times.

Since RLP is NP-Hard, such formulations cannot be used to solve realistic size problems and therefore we resort to discussions of heuristic approaches.

III. Simulated Annealing Algorithm

Simulated Annealing is an optimisation algorithm which originates from the physical simulation of a gaseous state being slowly heated and cooled into a solid state. Simulated Annealing tries to overcome the possibility of falling into local optima by simulating the physical cooling behaviour. This is done by using a random element in conjunction with the temperature parameter to generate a probability curve which determines when to accept a bad move in anticipation of a better convergence afterwards [22].

The following pseudo-code represents the general Simulated Annealing algorithm for a minimisation problem with solution space S, objective function f and neighborhood structure N.

```
Simulated_Annealing ( )
{   Select an initial solution S0;
Select an initial temperature t0 > 0;
Select a temperature reduction function α;
While (Not Freezing)
{   Initialise (Iteration_Count);
While (Iteration_Count <= nrep)
{   Randomly select s ∈ N(S0);
    σ = f(S) − f(S0);
    If (σ < 0) S0 = S;
    Else
    {Generate random x uniformly in (0, 1)
        if( x < e^(−σ/t) ) S0 = S; }
    Increment (Iteration_Count ); }
    Set t = α(t);
    Update Freezing; }
S0 is the approximate solution.
} // End Algorithm //
```

Figure 1: General Simulated Annealing Algorithm

In adapting this general algorithm to RLP, a number of general and specific decisions need to be made. The general decisions are concerned with the parameters of the algorithm itself i.e. initial temperature and cooling schedule. Problem specific decisions include the choice of the space of feasible solutions, neighbourhood structure and the form of the cost function.

A. Selection of the initial temperature (T_0)

In order for the final solution to be independent of the initial solution, T_0 must be 'hot' enough to allow a free exchange of neighbouring solutions.

Definition - The acceptance ratio χ is the ratio between the numbers of moves accepted by the algorithm to the total number of moves generated by the algorithm for a given temperature.

Initially, the value of χ should be close to one. That is, T_0 must be high enough such that virtually all generated moves are accepted. A more natural procedure could be the following, as proposed by Colin and Reeves [22];

The value of χ is fixed to a high value and then the system is heated (increase temperature) until the probability of acceptance ($x = e^{\frac{-\sigma}{T}}$) reachesχ. At this point, T can be taken as the initial temperature for the tested problem. This is analogous to the physical cooling process where a substance is heated quite rapidly to its liquid state before being cooled slowly to its ground state (thermal equilibrium).

However, we found that this procedure was too time-consuming as it entails the repetition of the stated process for each problem before cooling can commence.

We applied the formula below, which showed better results than the previously stated method.

$$T_0 = \frac{-\left(C_0 / T_f\right)}{\ln(\chi_0)}$$

Where; C_0 = Initial cost value,

T_f = the final value of temperature i.e. freezing point (0.01 was the best in this application).

The formula is derived as follows;

Since the probability of acceptance is given by the formula, $x = e^{\frac{-\sigma}{T}}$, then, $T = \frac{-\sigma}{\ln(x)}$;

If we assume that the probability of acceptance at the initial stage of the algorithm is χ_0, we have $T_0 = \frac{-\sigma}{\ln(x_0)}$;

We want to substitute σ with the maximum value of cost difference in the problem i.e. the difference between the maximum and minimum values of cost. Since this value is not known, we estimate it by multiplying the initial cost (C_0) by a factor greater than one, which we have chosen to be $\frac{1}{T_f}$ i.e. $\sigma = \frac{C_0}{T_f}$.

B. Cooling schedule

The rate at which temperature is reduced is vital to the success of the algorithm. This is dependent on the number of iterations at each temperature (*nrep*) and the temperature reduction function $\alpha(t)$.

Theory suggests that the system should be allowed to move very close to its stationary distribution at the current temperature before temperature reduction. However, the theory also suggests that this requires an exponential number of repetitions [22], and therefore it is necessary to find a way of reducing the number of iterations. This may be achieved by applying a large number of iterations at few temperatures or a small number of iterations at many temperatures.

One of the most widely used schedules involves a geometric reduction function;

$$\alpha(t) = at \text{ , where } a < 1$$

From experience, the most successful values of a are between 0.8 and 0.99 with a bias to the higher end of the range. The algorithm stops when the change in temperature is less than a given value (0.01 was better in this application).

However this purely theoretical condition is often tempered with more pragmatic factors such as the amount of time spent by the algorithm, or the amount of time since the last most optimum solution was discovered. The reason for taking this approach is that, the algorithm may continue searching for a long time after the stable optima has been achieved. Counting the number of times which the solution is repeated without any change we can check if this number exceeds a selected value (m^3 was chosen, where m is the length of the planning horizon) and stop the algorithm.

C. Number of repetitions at each temperature (nrep)

It is important to spend more time at lower temperatures so as to ensure that a local optimum has been fully explored. This can be done in various ways including the following two;

1. Starting with a small value of *nrep* at high temperature, we increase this value arithmetically by a small factor (0.05 chosen for this application) at each new temperature.
2. Feedback from the annealing process is used by accepting a fixed number of moves before decreasing temperature. This implies that a short time will be spent at high

temperatures when the rate of acceptance is high. Because the number of acceptances at very low temperatures will be low, it may take an infeasible amount of time to reach the required total, and it is therefore necessary to impose a maximum number of iterations per temperature as well (i.e. m^3).

Another commonly used temperature reduction function as suggested by Lundy and Mees [23], executes one iteration at each temperature, but reduces temperature very slowly according to the formula;

$\alpha(t) = \frac{t}{1+\beta t}$, where β is a suitably small value (β=0.01 was picked for this study after experimentation).

Data were collected for both functions and compared the results.

D. Solution space

Solution is a vector of starting times of each activity which corresponds to the near optimal even distribution of resource levels in the resource profile. Initial solution is a vector of the Earliest starting times of each activity. Since each activity can be scheduled anywhere within its 'total floating time' without affecting the completion time of the schedule, then the 'total solution space' is the set of all feasible combinations of these activity starting times over their total floats.

'Free floating time' (*FreeF*) of an activity is a part of its total floating time in which an activity can be scheduled without affecting the schedule of other activities in the activity network. The role of the total and free float in determining a feasible solution can be explained in terms of two general rules [1];

1. If the total float equals the free float, the non-critical activity can be scheduled anywhere between its earliest starting and latest completion times.
2. If the free float is less than the total float, the starting of the non-critical activity can be delayed relative to its earliest start time by no more than the amount of its free float without affecting the schedule of its immediately succeeding activities.

Thus if an activity's starting time does not fall into a free float range, the immediately succeeding activities must be pushed forward by at least the amount of deviation of this activity from the free float region (i.e. ∇ as shown in Figure 2).

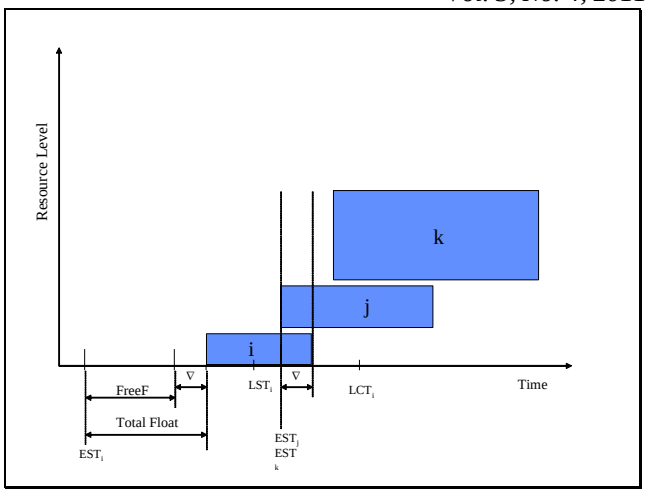


Figure 2: The effect of activity i on its immediate successors j and k.

In Figure 2, activities *j* and *k* are immediate successors of activity *i* i.e. *j* and *k* can start as early as *i* is completed. If *i* starts anywhere within the *FreeF* region, then it will always be completed before EST_j and EST_k which are the earliest starting times of activities *j* and *k* respectively. Activity *i* deviates from the *FreeF* by an amount ∇ and thus activity *j* must be pushed forward by at least the same amount ∇. It is noted that, we only push those activities that will otherwise violate the precedence relations. For example, activity *k* does not need to be pushed forward because it starts at a point beyond the completion of activity *i*.

E. Variable Neighbourhood structures

Variable Neighborhood Search is a method that is based on the systematic change of the neighborhood during the search. This is based on general observations that;

- a local minimum with respect to one neighbourhood structure is not necessarily locally minimal with respect to another neighbourhood structure
- a global optimum is locally optimal with respect to all neighbourhood structures

As a control mechanism we start with a fixed neighbourhood. Since the aim is to minimise the highest resource level in the resource profile, it seems reasonable to concentrate our search on the neighbourhood of this highest point. At each iteration, we find the highest point and search for the next solution by selecting random starting times for the activities which lies in the neighbourhood of its corresponding time (High) by a predetermined distance (*NbrHood*). That is, those activities which lie in the interval [*High-NbrHood*, *High+NbrHood*] as shown in Figure 3 provided that this range is feasible. This range is not feasible if;

- (*High* - *NbrHood*) < 0 in which case 0 is selected as the lower bound of the range.
- (*High* + *NbrHood*) > *m*, in which case m is selected as the upper bound of the range.

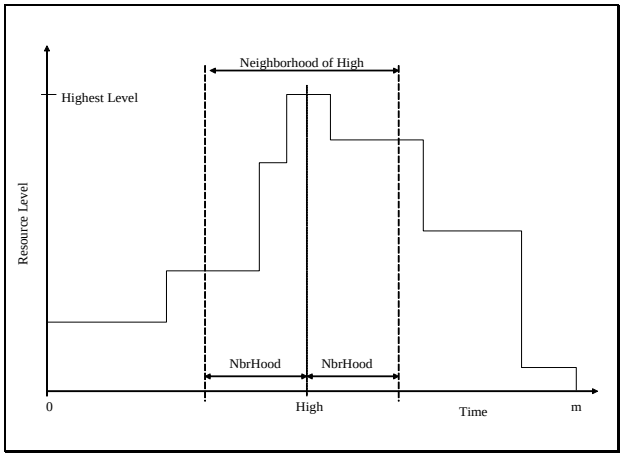


Figure 3: Neighbourhood structure

This structure was found to work well only on the cases where there was a guarantee that one High point will not dominate the search. That is, if the algorithm could move freely through the planning horizon. However this is not always the case. If at the beginning, a certain selected neighbourhood fails to drop below the next highest point which is not in its neighbourhood, then the algorithm will be stuck on this neighbourhood (see Figure 4).

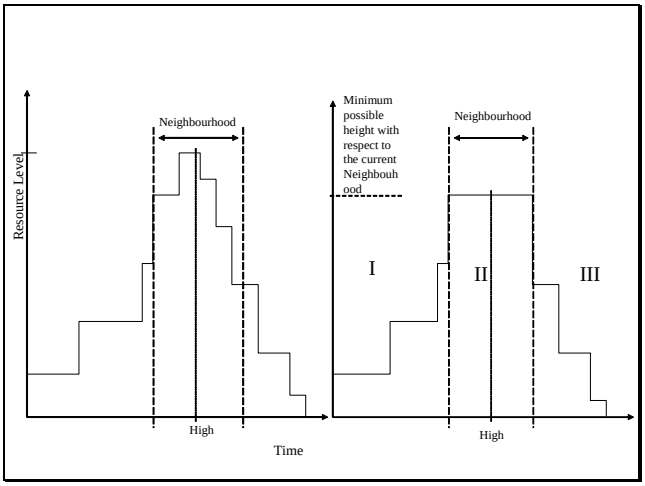


Figure 4: Limitations of the fixed neighbourhood structure

Surely, the movement of activities in regions I and II may result in more freedom of movements for activities in the current neighbourhood and lead to a better solution. To overcome this drawback, a second strategy is implemented as explained next.

1) Increasing Neighbourhood.

It is started with a small neighbourhood interval and slowly increases its value with decrease in temperature to a maximum value of m. Clearly, this is a more expensive strategy in terms of time, but it enables the algorithm to search in a wider space when the highest point is about to reach its stable position. That is, at each new temperature, we increase the searching space so as to increase the chances of an improvement in the solution at hand.

2) Decreasing Neighbourhood.

In Hanan and Bruce [24], it is suggested that large moves should be made at high temperatures so as to allow the algorithm to avoid local minimum, but as the temperature is lowered, smaller moves should be made to support the convergence to an equilibrium point. We implement this case by varying the neighbourhood interval from m to a minimum chosen value (2.0 was chosen after experimentation). This allows the algorithm to start by examining all activities within the planning horizon and then narrow the search as High moves to a more stable position. In this way, we minimise the possibility of being prematurely stuck on a fixed neighbourhood structure.

3) Cost function

One of the ways of detecting the most even distribution is to minimise the sum of the squares of the activity levels in the resource profile [11]. The resource level profile is divided into unit intervals, and finds the sum of squares of the resource levels over all units. The approximate solution corresponds to the minimum of these sums.

IV. GENERATION OF THE RESOURCE LEVELLING PROBLEMS

In order to test the algorithm, problems with varying features are needed. These are generated by considering the general problem ideas from manufacturing industry. In manufacturing industry, the scheduling process involves the allocation of the available capacity or resources (equipment, labour, and space) to jobs, activities, tasks, or customers through time [25]. Important documents required in this process include the Bill of Material (BOM) and the Master Production Schedule (MPS). One of the applications of both these documents (BOM and MPS) is to develop a resource profile by the work centre (or facility). This is used to determine whether the given orders can be delivered within the given time scale (Planning horizon) without exceeding the available resources. Essentially, BOM is developed into a tree structure which is a hierarchy of activities according to their precedencies. The BOM tree is converted into an activity network and then solved by the Critical Path Method (CPM) to get the initial profile.

BOM trees of varying distributions are generated and governed by the following features;

- Branching Factor (BF): This factor represents the number of branches emanating from each new parent node. In order to avoid the uniformity of branches on each parent node, we represent BF in terms of a range of values in which the number of branches for a particular parent node can fall. Thus different ranges of BF correspond to different distributions of nodes in the BOM tree.
- Number of Levels: This determines the height of the BOM tree and its variation will result in different distributions of BOM nodes.
- Duration and Resource Levels of each node are also generated at random at specified upper and lower bounds.

V. EXPERIMENTAL RESULTS

The algorithm was implemented in C++ and run on an Intel® 3 GHz CPU. The first digit in each of the names of problems listed stands for the number of levels and the last two digits stands for the branching factor. For example the problem C3_04 has 3 levels and a branching factor range of [0–4]. Two problems were generated for each set of parameters e.g. problems C3_04A and C3_04B. Nine cases were tested as explained in Table I together with the abbreviations used to represent each case.

TABLE I THE TESTED CASES WITH THEIR ABBREVIATIONS

Case	Reduction function	Iterations (*nrep*)	Neighbourhood	Abbreviation
1	Geometric (G)	Variable (V)	Fixed (F)	GVF
2	Geometric	Variable	Increasing (I)	GVI
3	Geometric	Variable	Decreasing (D)	GVD
4	Geometric	Fixed	Fixed	GFF
5	Geometric	Fixed	Increasing	GFI
6	Geometric	Fixed	Decreasing	GFD
7	Lundy & Mees (L)	10	Fixed	L1F
8	Lundy & Mees	10	Increasing	L1I
9	Lundy & Mees	10	Decreasing	L1D

Figure 5 shows graphical representation of the profile of one of the problems as randomly generated. The example presented (C3_04B) is generated from 3 orders each with a branching factor in the range [0, 4].

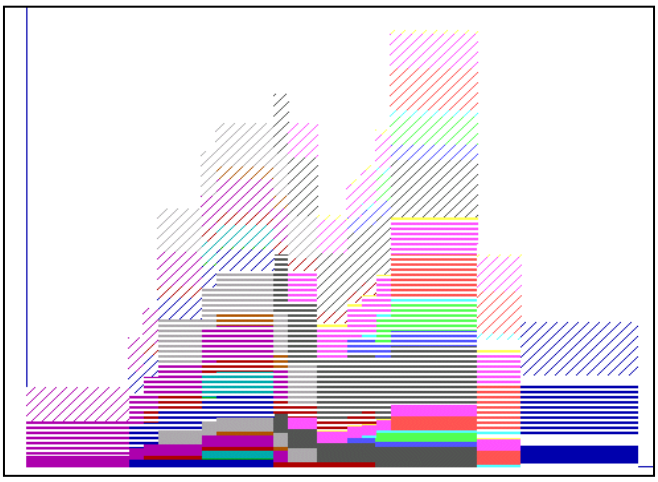


Figure 5: The initial Resource Profile for problem c3_04B.

EACH ORDER IS A BOM TREE WITH 18 NODES, AND THUS THE PROFILE PRESENTED IN FIGURE 5 IS MADE FROM 54 NODES OF THE 3 ORDERS AS SHOWN ON THE SOLUTION IN

Table II.

TABLE II PERFORMANCE OF THE ALGORITHM BY CASES

			Order 1		Order 2		Order 3	
Activity	Duratn	Init Start	Resrce Level	Final Start	Resrce Level	Final Start	Resrce Level	Final Start
1	0	0	0	11	0	11	0	11
2	10	0	7	7	21	7	21	7
3	5	0	0	12	0	12	0	12
4	5	0	7	12	21	12	21	12
5	16	0	4	8	12	8	12	8
6	18	0	11	2	33	3	33	1
7	5	0	4	7	12	7	12	12
8	11	0	14	3	42	9	42	4
9	14	10	19	17	57	17	57	17
10	9	0	5	22	15	22	15	22
11	7	16	9	24	27	24	27	24
12	11	0	2	14	6	3	6	1
13	9	0	14	15	42	3	42	24
14	16	18	11	21	33	19	33	19
15	14	11	0	13	0	20	0	17
16	11	24	3	30	9	32	9	29
17	8	34	17	34	51	36	51	35
18	0	42	0	45	0	43	0	44

The Starting Cost : 2961511
The Final Cost : 2549043
Planning Horizon : 45, CPU Time (Sec) : 21

Figure 6 shows the generated final profile, which represent the best solution found from the GVI case.

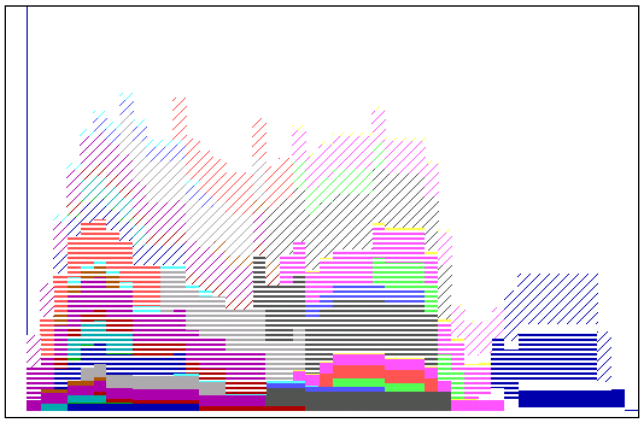


Figure 6: Final Resource Profile for the problem c3_04B by GVI.

Clearly, Figure 6 is more evenly distributed compared to Figure 5. It is also noted that the height of the profile has been reduced considerably. The difference between the initial and the final cost is 412,468 which is an improvement of 14%. The highest peak of this initial profile is 427 units while the highest peak of the final profile represented in Figure 6 is 328 units.

The performance of a solution is the percentage of improvement in cost value associated with that solution in relation to the cost value associated with the initial solution.

$$\text{i.e. } Performance = \left(\frac{\text{Initial cost - Final cost}}{\text{Initial cost}}\right)\times 100$$

Cases L1I and GVI showed significantly better results (performance) compared to the other cases as shown in Table III, which presents a sample of the tested problems with different properties.

TABLE III SUMMARY OF RESULTS FOR PROBLEM C3_04B.

Problem	GVF	GVD	GVI	GFF	GFD	GFI	L1F	L1D	L1I
c3_02A	6%	16%	36%	19%	22%	33%	16%	18%	36%
c3_02B	14%	6%	17%	15%	13%	16%	13%	15%	17%
c4_03B	2%	7%	15%	2%	15%	13%	2%	5%	15%
c2_04A	15%	15%	22%	6%	22%	22%	15%	19%	22%
c2_04B	10%	15%	24%	10%	22%	10%	9%	12%	23%
c4_04A	0%	20%	27%	5%	26%	15%	2%	17%	27%
c4_04B	27%	33%	35%	20%	35%	32%	35%	27%	35%
c5_04A	2%	35%	36%	2%	30%	17%	2%	24%	36%
c5_04B	0%	19%	19%	0%	19%	9%	2%	10%	19%
c4_05B	12%	21%	26%	8%	21%	18%	8%	14%	27%
c5_05A	11%	23%	27%	11%	27%	25%	13%	18%	27%
c5_05B	10%	42%	44%	13%	42%	1%	11%	27%	43%
c3_06A	10%	7%	26%	7%	26%	21%	11%	12%	26%
c3_06B	1%	5%	19%	1%	18%	16%	1%	12%	19%
c4_06A	8%	36%	36%	11%	36%	28%	12%	25%	36%
c4_06B	19%	17%	28%	12%	27%	22%	19%	15%	28%

As noted in Table III, both successful neighbourhood structures are variable and specifically increasing order. Increasing neighbourhood implies that the search space is widened as we approach convergence, making it possible to explore wider ranges and therefore increase the chances for a better profile.

ALL CASES THAT INVOLVED FIXED NEIGHBOURHOOD PERFORMED POORLY (GVF, GFF, L1F). THE CHOICE OF *NREP* APPEARS TO HAVE NO SIGNIFICANT ROLE IN THE PERFORMANCE OF THE ALGORITHM, SINCE THE BEST STRATEGIES INCLUDED BOTH VARIABLE AND FIXED NUMBER OF ITERATIONS (*NREP*). ALL PROBLEMS WERE SOLVED WITHIN A FEW SECONDS AS SHOWN IN THE SAMPLE SOLUTION IN

Table II.

VI. CONCLUSION

Variable neighborhood provides considerable improvements when applied to global heuristic techniques for the Resource Levelling Problem. Increasing neighbourhood structure allows for higher exploration of the resource profile; since the final iterations of Simulated Annealing algorithm have shown to perform better than decreasing neighbourhood. Simulated Annealing is one of very successful global heuristic techniques and has performed well in this particular implementation. However, Simulated Annealing is known to depend very much on fine-turning of parameters for its performance. The selected parameters for this application therefore have shown to perform well, although there is always a room for improvement.

Further Research

A total of 44 problems were tested, which were generated randomly. More problems with additional varying features may lead to a better and clearer picture about characteristics of the problem. For instance, in the construction of orders we assumed that each order involves the whole BOM tree. This is not always the case, and one may wish to incorporate this fact into the algorithm.

REFERENCES

[1] H., A. Taha, "Operations Research: An Introduction", Macmillian Publishing Company, USA, 1992.

[2] K., Neumann, C., Schwindt, and J. Zimmermann., "Project Scheduling with Time Windows and Scarce Resources", Springer, Berlin, 2003.

[3] J. D., Stevens "Techniques for construction network scheduling". McGraw-Hill, New York, 1990.

[4] J. Martinez, and P. Ioannou "Resource leveling based on the modified minimum moment heuristic", Proc.,5th Int. Conf., Computing in Civ. And Bulid. Engrg., ASCE, Reston, Va.,287-294, 1993.

[5] J. Son and M., J. Skibniewski "Multiheuristic approach for resource leveling problem in construction engineering: hybrid approach", Journal of construction engineering and management, 125(1), 23-31, 1999.

[6] J. D. Wiest and F. K. A. Levy. A Management guide to PERT/CPM: with GERT/PDM/DCPM and other Networks, New Delhi, Prentice-Hall of India, 1977.

[7] J. J. Moder, C. R. Philip and E. W. Davis, Project Management with CPM, PERT and Precedence Diagramming, 3rd edn, Von Nostrand Reinhold, 1983.

[8] R. V. Galbreath, "Computer program for leveling resource usage", Journal of the Construction Division, ASCE, 91(1), pp. 107–124, 1965.

[9] R. B. Harris, "Packing method for resource leveling", Journal of Construction Management, 61(2), pp. 331–351, 1990.

[10] R. B. Harri, Precedence and Arrow Network for Construction, John Wiley, 1978.

[11] A. R. Burgess and J. R. Killebrew, "Variation in activity level on a cyclical arrow diagram", Journal of Industrial Engineering, 13(2), pp. 76–83, 1962.

[12] V. A. Jeetendra, O. V. C. Krishnaiah and J. R. Prashanth, "Petri Nets for Project Management and Resource Levelling", International Journal of Advanced Manufacturing Technology, Springer-Verlag, 16, pp. 516-520, 2000.

[13] Z. Sheng-li, L. Yan, Z. Hong-Mei, Z. Ri-Lin, "GA-based Resource Leveling Optimization for Construction Project", Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, IEEE, 13-16 August 2006, pp. 2363-2367.

[14] B. Ahmed, N. E. Neil, "Use of Genetic Algorithms in Resource Scheduling of Construction Projects", Journal of Construction and Engineering Management, 2004, 130(6), pp. 869-877.

[15] M. Malek, H. Alain, "A GA-Based Fuzzy resource levelling optimization for helicopter maintenance activity", EUSFLAT-LFA, Atlantis Press, 2011, pp. 667-672.

[16] X. Ying, K. Ya Ping, "Ant Colony Optimization Algorithm for Resource Leveling Problem of Construction Project", The CRIOCM 2006 International Symposium on "Advancement of Construction Management and Real Estate", 2006.

[17] Q. Jian-Xun, W. Qiang, G. Xin-Zhin, "Improved Particle Swam Optimization For Resource Levelling Problem", Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, IEEE, 2007, pp. 896-901.

[18] S. Ivan, P. Marko, S. Predgrag, C. Miroslav, "Heuristic Algorithm for Single Resource Constrained Project Scheduling Problem Based on the Dynamic Programming", Yugoslav Journal of Operations Research, 19(2), 2009, pp. 281-298.

[19] J. Zimmermann, J. H. Bartels, "Tree-based Methods for Resource Investment and Resource Levelling Problems", in K. Gurlebeck and C. Konke (eds.), Proceedings of the 17th International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering, Weimar, Germany, 12–14 July 2006

[20] A. Kastor, K. Sirakoulis, "The effectiveness of resource levelling tools for Resource Constraint Project Scheduling Problem", International Journal of Project Management, 27 (2009), pp. 493–500.

[21] A. R. Mushi., M. O'heigeartaigh, "The Resource Levelling Problem (RLP) Mixed Integer Programming Formulations, Uhandisi Journal, 23(1), pp. 13-20, 1999.

[22] R. Colin, Reeves (Ed.), "Modern Heuristic Techniques for Combinatorial Problems", Blackwell Scientific Publications, Oxford, 1993.

[23] M. Lundy M., A. Mees, "Convergence of an Annealing Algorithm". Math. Prog. , 34, 111-124, 1986.

[24] L., Hanan, M., Bruce, "Composite Stock Cutting through Simulated Annealing", University of Missouri at Rolla, Research Report Numbers CSC 91-09 & ISC 91-04, 1991.

[25] G. S. Roger, "Operations Management, Decision Making in the operations function", McGraw - Hill, Inc., New York, 1993.