An Iterated Greedy Algorithm for the Node Placement Problem in Bidirectional Manhattan Street Networks

Fubito Toyama
Faculty of engineering,
Utsunomiya University
7-1-2, Yoto, Utsunomiya-shi,
321–8585 JAPAN
fubito@is.utsunomiya-u.ac.jp

Kenji Shoji
Faculty of engineering,
Utsunomiya University
7-1-2, Yoto, Utsunomiya-shi,
321–8585 JAPAN
shoji@is.utsunomiya-u.ac.jp

Juichi Miyamichi
Faculty of engineering,
Utsunomiya University
7-1-2, Yoto, Utsunomiya-shi,
321–8585 JAPAN
miya@is.utsunomiya-u.ac.jp

ABSTRACT

Wavelength Division Multiplexing (WDM) is a technology which multiplexes optical carrier signals on a single optical fiber by using different wavelengths. Lightwave networks based on WDM are promising ones for high-speed communication. If network nodes are equipped with tunable transmitters and receivers, a logical topology can be changed by reassigning wavelengths to tunable transceivers of nodes. Network performance is influenced by the logical node placements. Therefore, an efficient algorithm to obtain the optimal node placement to achieve the best network performance is necessary. In this paper, an iterated greedy algorithm is proposed for this node placement problem. The proposed iterated greedy algorithm consists of two phases, construction and destruction phases. As a local search algorithm, variable depth search is applied after the construction phase. The computational results showed that this iterated greedy algorithm outperformed the best metaheuristic algorithm for this problem.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Metaheuristics, Iterated greedy

1. INTRODUCTION

Recently, the Internet is used in many fields and is growing rapidly. Internet users are explosively increasing and the Internet is a main part of our life. The network capacity is not sufficient because the communications traffic is increasing year by year. Therefore expanding network capacity has become a demand. Wavelength Division Multiplexing (WDM) is a technology which multiplexes optical carrier signals on a single optical fiber by using different wavelengths. WDM are promising ones for wideband communication networks. There are two types of WDM networks, single-hop and multi-hop networks [11, 12]. In single-hop networks, source and destination nodes communicate directly without any intermediate nodes. Single-hop networks achieve higher throughput than multi-hop networks. On the other hand, in multi-hop networks, each network node is equipped with a limited number of transmitters and receivers. Thus, source and destination nodes may communicate through intermediate nodes. The logical topology of multi-hop networks can be changed by reassigning wavelengths to the transmitters and receivers of the nodes. If new nodes are added to a system, a new network can be easily constructed by reassigning wavelengths, without requiring new hardware. Therefore multi-hop networks are suitable for constructing large scale networks because it is easy to expand. Multi-hop networks with regular topology are suitable for high-speed communication because the routing between the nodes is simple. Bidirectional Manhattan Street Network (BMSN), one of the multi-hop networks with regular topology, provides high performance compared with other networks with regular topology [10, 2, 1]. For this reason, we focus on the node assignment in the BMSN. In WDM-based multi-hop networks, since each node is equipped with tunable transmitters and receivers, the logical topology can be changed by reassigning wavelengths to tunable transceivers of nodes. In BMSN, each node has four sets of transmitters and receivers. Bidirectional direct links between one node and four different nodes are constructed by these transmitters and receivers. Figure 1 and figure 2 show an example of a BMSN physical topology and its logical topology of four network nodes (N_0, N_1, N_2, N_3). In figure 2, the logical topology of N_0, N_1, N_2, and N_3 is changed by reassigning wavelengths λ_0 to λ_{15}. Network performance is influenced by their node placements. Therefore, the efficient algorithm to obtain the optimal node placement to achieve the best network performance is necessary. For this problem, various metaheuristics such as multistart local search, tabu search, simulated annealing, and genetic algorithm are applied by Kato and Oie

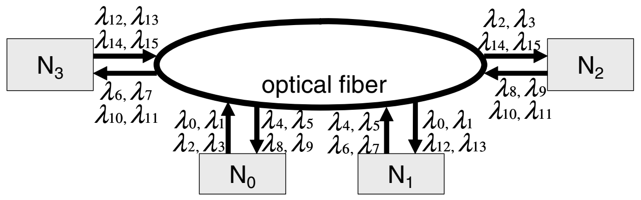


Figure 1: A BMSN physical topology.

[6]. They reported that the tabu search using an initial solution generated by a greedy method is the best one in their comparison. Kitani, et al. proposed a new simulated annealing approach called HIWAS (HIerarchical Wavelength ASsignment algorithm) in which an initial solution is generated hierarchically at first and the initial solution is improved by simulated annealing [7]. They showed that HIWAS obtains better solutions with shorter running times than the tabu search, but there is a restriction that the size of BMSN is fixed by $2^m \times 2^m$ (where m is a positive integer). Katayama, et al. proposed a new local search called variable k-swap local search (KLS) [5], based on variable depth search. In KLS, the neighborhood is defined as sequences of the single-swap moves and the length of the sequence is adaptively decided. Therefore, a small fraction of the large neighborhoods can be searched efficiently in reasonable times. They proposed iterated k-swap local search (IKLS), an iterated local search metaheuristic embedded with KLS. Their results showed that IKLS was highly effective when compared to state-of-the-art methods.

In this paper, an iterated greedy algorithm is proposed for the node placement problem. Iterated greedy (IG) has been applied to various optimization problems with success [14, 3, 9]. Although IG algorithm is very simple, it is very effective as shown in their experimental results. The proposed IG algorithm consists of two central phases, construction and destruction phases. These phases are applied in [14], and the performance is very competitive to the other metaheuristics. In the proposed method, variable k-swap local search (KLS) by Katayama, et al. [5] is applied after the construction phase as a local search algorithm. The computational results showed that our IG algorithm with KLS, called IGKLS, outperformed IKLS which is the best performing algorithm for this problem.

2. NODE PLACEMENT IN BMSN

We assume that the network consists of $X \times Y (= N_{size})$ nodes. Each node is equipped with four sets of tunable transmitters and receivers. In BMSN, bidirectional direct links between one node and four different nodes are constructed. The logical topology is symmetric and 2D torus. Figure 1 and figure 2 show an example of a 2×2 BMSN physical topology and its logical topology. In figure 2, Node N_0 can communicate directly with node N_1 and N_0. But node N_0 and N_3 communicate through intermediate node N_1 or N_2. Therefore, the total amount of traffic between N_0 and N_3 is twice as much as that of traffic between N_0 and N_1. The logical topology can be changed by reassigning wavelength to transmitters and receivers of the nodes. Network performance is influenced by the node placements. Therefore, the efficient algorithm to obtain the optimal wavelength assignment to achieve the best network performance is necessary. This wavelength assignment problem can be regarded as a node placement problem.

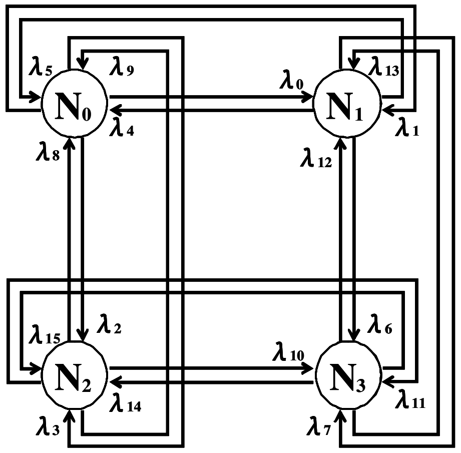


Figure 2: A BMSN logical topology of figure 1.

In this paper, the optimal node placement in which the total amount of traffic is minimized is searched. Once BMSN node placement is decided, wavelengths of all nodes can be assigned automatically.

In a $X \times Y$ BMSN physical topology, the logical address of the node on the k-th row and the l-th column is represented as (k, l) $(k = 0, 1, \ldots, X-1, l = 0, \ldots, Y-1)$. Figure 3 shows the logical address of the node in $X \times Y$ BMSN physical topology. $X \times Y$ nodes are assigned to these logical addresses in the node placement problem.

Network node placements are represented by the $X \times Y$ placement matrix. An element of placement matrix is a node number. For example, the node placement of figure 2 can be represented by matrix $\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$. Let $x_{(k,l)} \in \{0, 1, \ldots, N_{size} - 1\}$ denote a node number at place (k, l), and let σ denote the node placement matrix of $X \times Y$ whose element is $x_{(k,l)}$. Equation (1) represents an example of a node placement matrix. This matrix is generated from the logical topology of figure 4.

$$\sigma = \begin{bmatrix} 5 & 15 & 6 & 3 \\ 13 & 0 & 14 & 8 \\ 4 & 9 & 11 & 2 \\ 12 & 7 & 1 & 10 \end{bmatrix} \tag{1}$$

Furthermore, let $t_{i,j}$ denote the amount of traffic from node N_i to node N_j $(i \neq j)$, and $\mathbf{T} = [t_{i,j}](i, j = 0, 1, \ldots, N_{size} - 1)$ represents the traffic matrix of $X \times Y$ whose element is $t_{i,j}$. We assume that the traffic matrix $\mathbf{T}$ is known beforehand. The amount of traffic, $t_{i,j}$, is simplified into two types of traffic, heavy traffic flow, t_H, and light traffic flow, t_L. t_H and t_L are set to 1 and 0 respectively in our experiments. The total amount of traffic in the network given by σ is defined as

$$F(\sigma) = \sum_{i=0}^{N_{size}-1} \sum_{j=0}^{N_{size}-1} t_{i,j} \times h(d_i, d_j) \tag{2}$$

where d_i represents the logical place of a node N_i, and

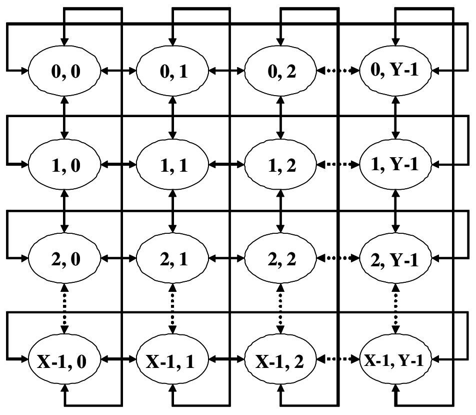


Figure 3: Logical address of the node in BMSN.

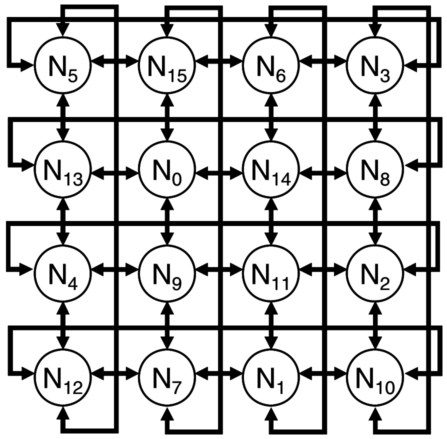


Figure 4: Example of node placements of a 4×4 BMSN.

$h(d_i, d_j)$ represents the number of hops along the shortest path between d_i and d_j. Let d_i, d_j be (i, j) and (k, l). $h(d_i, d_j)$ is defined as

$$h(d_i, d_j) = min(|i-k|, X-|i-k|) + min(|j-l|, Y-|j-l|). \tag{3}$$

For example, in the node placement matrix of equation (1), the number of hops between d_5 (place (0,0)) and d_8 (place (1,3)) is 2, i.e. $h(d_5, d_8) = 2$.

In this paper, the optimal node placement in which the total amount of traffic, $F(\sigma)$, is minimized is searched from the obtained traffic matrix $\mathbf{T}$.

3. AN ITERATED GREEDY ALGORITHM FOR THE NODE PLACEMENT PROBLEM

The goal of the node placement problem is to minimize the cost function F, expressed by equation (2). In this paper, an iterated greedy (IG) algorithm with variable k-swap local search, called IGKLS, is proposed for this node placement problem. IGKLS consists of two central phases, construction and destruction phases, and a local search. The outline of the IGKLS is shown in figure 5

```
Procedure Iterated_Greedy_for_NPP
begin
    $\sigma$ := Greedy_heuristic;
    $\sigma$ := LocalSearch($\sigma$);
    $\sigma_{best}$ := $\sigma$
    while (termination criterion not satisfied) do
        $\sigma_d$ := remove $d$ nodes at random from $\sigma$
        $\sigma$ := reassign $d$ nodes by greedy heuristic
        $\sigma$ := LocalSearch($\sigma$);
        if $F(\sigma) < F(\sigma_{best})$ then
            $\sigma_{best}$ := $\sigma$;
        endif
    endwhile
    return $\sigma_{best}$
end
```

Figure 5: An iterated greedy algorithm for the node placement problem.

First, an initial solution which denotes an initial node placement is generated by a greedy heuristic. Then the solution is locally optimized by a local search. Next, d nodes are removed at random from the locally optimal solution in the destruction phase. In the construction phase, these d nodes are reassigned to this solution by a greedy heuristic. In order to avoid generating the same solution, the removed node is not reassigned to the same place again. Then the new solution is locally optimized by a local search again. The destruction, construction, and local search steps are repeated until some stopping criterion is met. Finally, the best solution, σ_{best}, found in the search is returned.

3.1 Generating an initial solution

First, an initial solution which denotes an initial node placement is generated by a greedy heuristic. As a greedy heuristic for generating an initial solution, we use the greedy method proposed by Kato et al. [6].

Let U be the set of unassigned nodes. $u \in U$ represents an element of the set U, and let $f_u^{(h)}$ be the total amount of offered traffic between node u assigned at the specific place and each node already assigned at the places which are h hops away from the specific place. Then the greedy heuristic selects the node u which has the maximum value of the following evaluation function $z(u)$.

$$z(u) = \sum_{h=1}^{H} \alpha_h f_u^{(h)} \tag{4}$$

where α_h represents the weight of the evaluation function, and H represents the limited hops of the evaluation function. $\alpha_h = 1/2^{h-1}$ and $H = \sqrt{N_{size}}/2$ are set in our experiments. The first node u is selected randomly and assigned at the place (0,0). In step k, the node u which has the maximum value of the function (4) is assigned at the place ($\lfloor k/Y \rfloor$, k mod Y).

3.2 Local search

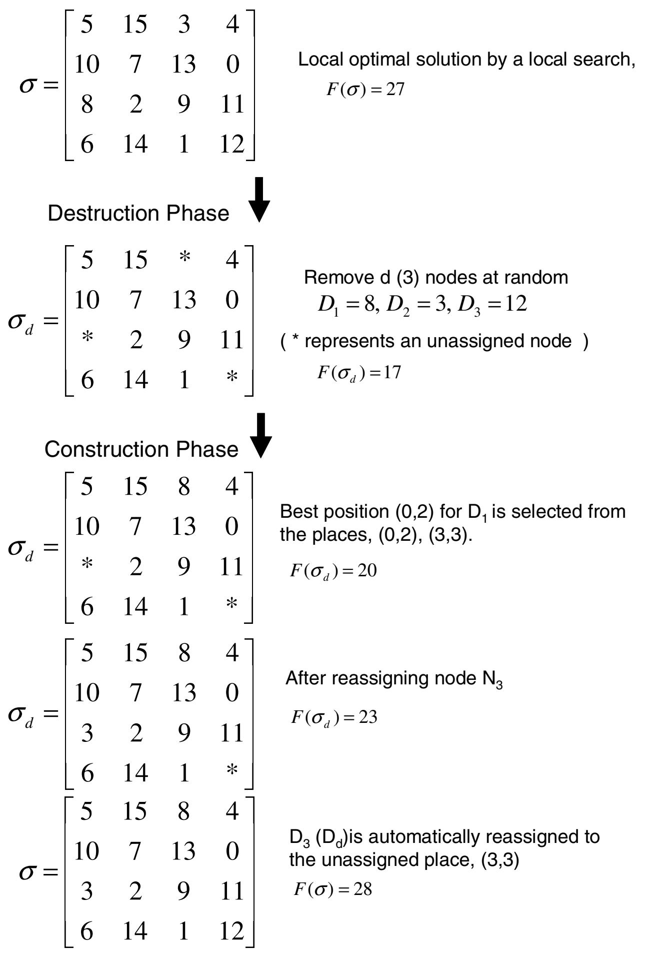
The performance of a local search depends on the definition of a neighborhood. One of the simplest local search methods is the 1-swap local search in which a neighborhood is defined as the set of solution derived by swapping a pair of nodes in the current solution. Katayama, et al. proposed a

```
Procedure Variable_k-swap_LS ($\sigma$)
begin
  $\sigma_{best} := \sigma$, $P_{all} := \{0, \ldots, N_{size} - 1\}$, $g_{LastImp} := \infty$;
  repeat
    $\sigma := \sigma_{best}$, $g := 0$, $g_{best} := 0$;
    $P_{part} := \{0, \ldots, N_{size} - 1\}$;
    $i_{base}$ := select a node at random from $P_{all}$;
    $P_{part} := P_{part} \backslash \{i_{base}\}$, $P_{all} := P_{all} \backslash \{i_{base}\}$;
    repeat
      find a node $i$ with
          $min_{i \in P_{part}} \delta_i := SwapGain(i_{base}, i, \sigma)$;
      $\sigma := SwapMove(i_{base}, i, \sigma)$;
      $g := g + \delta_i$, $P_{part} := P_{part} \backslash \{i\}$;
      if $g_{best} > g$ then $\sigma_{best} := \sigma$, $g_{best} := g$
    until $P_{part} = \emptyset$ or $g > g_{LastImp}$;
    if $g_{best} < 0$ then
        $P_{all} := \{0, \ldots, N_{size} - 1\}$, $g_{LastImp} := |g_{best}|$;
  until $P_{all} = \emptyset$;
  return $\sigma_{best}$
end
```

Figure 6: Variable k-swap local search (KLS).

variable k-swap local search (KLS) based on variable depth search (VDS). The idea of VDS was first proposed by Lin and Kernighan [8]. In VDS, the neighborhood is defined as sequences of moves to a given solution rather than only one move at each iteration. The length of the sequence is variable. Thus, the size of neighborhood is also variable. VDS can effectively traverse larger search space within reasonable time by changing the size of the neighborhood adaptively. Similarly, in KLS, the neighborhood is defined as sequences of the single-swap moves and the length of the sequence is adaptively decided. VDS and KLS were applied in many combinatorial optimization problems, and its effects were reported in [13, 15, 4].

Therefore, we use KLS as a local search algorithm for the node placement problem. The pseudo-code of KLS is shown in figure 6. In figure 6, P_{all} and P_{part} are used to avoid being swapped twice in one sequence. KLS consists of outer and inner loops. First, node i_{base} is selected randomly from P_{all}. In the inner loop, the counterpart node i for which the value of $SwapGain(i_{base}, i, \sigma)$ is minimal is selected from P_{part}, where $SwapGain(i_{base}, i, \sigma)$ is a cost difference between the current solution and the neighbor one by swapping nodes i_{base} and i. The pair of nodes i_{base} and i is swapped by $SwapMove(i_{base}, i, \sigma)$ even if the value of $SwapGain(i_{base}, i, \sigma)$ is larger than zero, i.e. the cost function $F(\sigma)$ is worse. Such swapping is repeated until $P_{part} = \emptyset$. In the inner loop, the best gain value g_{best} and its solution σ_{best} are saved. If the best gain value is smaller (better) than zero, P_{all} is initialized. In the outer loop, the selection of i_{base} and inner loop is repeated until $P_{all} = \emptyset$. Finally, the best solution σ_{best} is returned. To reduce the running time, $g_{LastImp}$ which is the best gain value recorded at the previous last iteration is used in figure 6. The inner loop is terminated if the current gain is larger than $g_{LastImp}$. By adding this termination condition, the efficiency of KLS increases without large loss of solution qualities. The details of KLS is described in [5].



$\sigma = \begin{bmatrix} 5 & 15 & 3 & 4 \\ 10 & 7 & 13 & 0 \\ 8 & 2 & 9 & 11 \\ 6 & 14 & 1 & 12 \end{bmatrix}$
Local optimal solution by a local search,
$F(\sigma) = 27$
Destruction Phase
$\sigma_d = \begin{bmatrix} 5 & 15 & * & 4 \\ 10 & 7 & 13 & 0 \\ * & 2 & 9 & 11 \\ 6 & 14 & 1 & * \end{bmatrix}$
Remove d (3) nodes at random
$D_1 = 8, D_2 = 3, D_3 = 12$
(* represents an unassigned node)
$F(\sigma_d) = 17$
Construction Phase
$\sigma_d = \begin{bmatrix} 5 & 15 & 8 & 4 \\ 10 & 7 & 13 & 0 \\ * & 2 & 9 & 11 \\ 6 & 14 & 1 & * \end{bmatrix}$
Best position (0,2) for D_1 is selected from the places, (0,2), (3,3).
$F(\sigma_d) = 20$
$\sigma_d = \begin{bmatrix} 5 & 15 & 8 & 4 \\ 10 & 7 & 13 & 0 \\ 3 & 2 & 9 & 11 \\ 6 & 14 & 1 & * \end{bmatrix}$
After reassigning node N_3
$F(\sigma_d) = 23$
$\sigma = \begin{bmatrix} 5 & 15 & 8 & 4 \\ 10 & 7 & 13 & 0 \\ 3 & 2 & 9 & 11 \\ 6 & 14 & 1 & 12 \end{bmatrix}$
D_3 (D_d)is automatically reassigned to the unassigned place, (3,3)
$F(\sigma) = 28$

Figure 7: Example of the destruction and the construction procedures.

3.3 Destruction and construction phases

The destruction and the construction procedures are two central procedures in IGKLS. The destruction procedure is applied to a locally optimal solution σ obtained by KLS. First, d nodes are selected randomly. These d nodes are then removed from σ in the order in which they were selected. Let σ_d denote a node placement after the removal of the d nodes. The cost value of σ_d is calculated by equation (2). However, the number of hops including d nodes is set to 0 in the calculation of equation (2), i.e. the value of (h_i, h_j) including the removed nodes is set to 0.

In the construction procedure, d nodes are reassigned to σ_d in the order in which they were removed from σ. Let D_i ($i = 1, 2, \ldots, d$) denote the removed d nodes, where i represents the order in which they were selected. First, D_1 is reassigned to all unassigned places except for the place before the removal of the node D_1. The cost value is calculated for these places, The cost function is the same one that is used in the destruction procedure. Finally, D_1 is reassigned the best place (the smallest $F(\sigma_d)$). This process is iterated until D_{d-1} is reassigned. The last removed node D_d is automatically reassigned to an unassigned place without the calculation of $F(\sigma_d)$.

Table 1: Performance comparison of IGKLS and IKLS

Method	N_{size}	F	Q	Time(s)	#Iter
IKLS	16	19.00	1.0000	$< \epsilon$	2.05
	64	82.25	0.9240	0.14	46.10
	256	366.20	0.8383	15.92	206.90
	1024	1681.55	0.7303	3230.30	874.70
IGKLS	16	19.00	1.0000	$< \epsilon$	1.90
	64	83.35	0.9125	0.13	27.15
	256	358.55	0.8565	15.50	162.50
	1024	1533.15	0.8012	2201.63	829.90

An example of the destruction and construction procedures using $d = 3$ is illustrated in figure 7. In this example, the d nodes ($d = 3$), N_8, N_3 and N_{12}, are first removed from σ. The cost value of σ_d is calculated. The traffic amount for the d nodes is not included in the cost function. Thus, $F(\sigma_d)$ is smaller than the cost value before the removal of nodes. In the construction phase, the best place $(0, 2)$ for D_1 (node N_8) is selected from all unassigned places, $(0, 2)$ and $(3, 3)$, except for the place $(2, 0)$, and N_8 is reassigned to the place $(0, 2)$. Similarly, D_2 (node N_3) is reassigned to the place $(2, 0)$. The last removed node D_d (N_{12}) is automatically reassigned to the remaining unassigned place $(3, 3)$. The same place before the removal of the node D_d can be selected in the reassignment of D_d. The new solution generated by the destruction and construction procedures is locally optimized by the local search (KLS) as shown in figure 5.

4. EXPERIMENTAL RESULTS

To evaluate the performance of IGKLS, we compare the performance of IGKLS with that of IKLS proposed by Katayama, et al. [5]. IKLS is currently the best metaheuristic algorithm for the node placement problem. For the comparisons, the benchmark instances provided by Katayama [5] are used. The benchmark set consists of 80 instances of 4 different problem sizes, $N_{size} = 16$ (4×4), 64 (8×8), 256 (16×16), 1024 (32×32). 20 different traffic matrices are contained for each problem size.

In the benchmark set, the traffic matrices are generated by assigning two types of traffic flow, heavy traffic flow,t_H, and light traffic flow, t_L. t_H and t_L are set to 1 and 0 respectively. The elements of the traffic matrices are only 0 (t_L) or 1 (t_H). Each node has four outgoing links in BMSN. Therefore the total number of outgoing links in BMSN is equal to $4N_{size}$. The number of $\lfloor 4aN_{size} \rfloor$ ($0 < a < 1$) links are randomly selected among $4N_{size}$ outgoing links, and heavy traffic flows are assigned to the selected links. The number of the outgoing links assigned heavy traffic flows is limited to less than L_{max} ($1 \leq L_{max} \leq 4$) per node. Given parameters a and L_{max} are set to 0.3 and 3 respectively in the benchmark. Light traffic flows are assigned all the other pairs of nodes. In the traffic matrices (problem instances) generated by this process, the optimal solution and its value of $F(\sigma_{opt})$ are known beforehand. For example, if the optimal node placement in 4×4 BMSN is defined as

$$\sigma_{\mathbf{opt}} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \tag{5}$$

the candidates of heavy traffic flows are the links between neighbors in σ_{opt} such as the links from N_0 to N_1, N_9 to N_5, N_{14} to N_2, and N_4 to N_7. The other elements (except heavy traffic) of the traffic matrix is set to 0 (light traffic). In σ_{opt}, the number of hops between all the heavy traffic nodes is 1, and the minimum value of the total amount of traffic F is the number of heavy traffic links.

The proposed IG algorithm was coded in C, and run on Intel Xeon 3.0 GHz PC, 2GB RAM, using the gcc compiler 2.95 with '-O3' option. We set the parameter $d = 2\sqrt{N_{size}}$ (d is used in the destruction phase). Therefore, the users parameter settings for each of the problem size, are not required in IGKLS.

The performance of IGKLS was compared with IKLS. The iteration number of each method (the execution number of local searches) is set to N_{size}. Thus, the two methods are executed in almost the same condition. Table 1 shows the results of IGKLS and IKLS. The results of IKLS were cited from [5]. IKLS algorithm has been coded in C, and the results have been obtained on Pentium4 3.4GHz, 4GB RAM, using the gcc compiler 4.11 with '-O3' option. Although the computer systems are different, the iteration number of local searches of each method is the same parameter (N_{size}). Furthermore, the same local search algorithm (KLS) is used in IGKLS and IKLS methods. Therefore, we focus on the quality of obtained solutions rather than the running time. The first two columns of the table 1 represent the name of the method and the problem size N_{size}, respectively. In the following columns we show the average cost value (F) of the best solutions obtained in each of 20 instances, its quality (Q), the average running time (Time(s)) in which the algorithm found the best solution, and the average number of local searches to the best solution. The solution quality Q (with the range $[0, 1]$) is defined as $Q = f(\sigma_{opt})/f(\sigma)$, where σ_{opt} is the optimum solution. $Q = 1.0$ means the obtained solution σ is the optimum. "$< \epsilon$" in Times(s) means that the average running times are less than 0.01 seconds.

The results indicate that the solution quality Q of IGKLS is better than that of IKLS for the larger size of problems. Especially, for the largest problem ($N_{size} = 1024$), IGKLS performs much better than IKLS. For the small size problem, the performance of IGKLS is almost the same as that of IKLS, although the performance of IGKLS is a little worse than that of IKLS for the problem size $N_{size} = 64$. The value of #Iter of IGKLS also obtained better results. Therefore, the average running time of IGKLS is also better than IKLS, despite using a slower CPU (in terms of clock speed) in our method.

Next, we compare the running times and 1-swap move

Table 2: Comparison of running times to reach the specified solution quality

N_{size}	$Q(F)$	IGKLS	IKLS
		Time(s)	Time(s)
16	1.0(19.00)	$< \epsilon$	$< \epsilon$
64	0.8(95.00)	$< \epsilon$	$< \epsilon$
256	0.7(438.57)	0.40	0.52
1024	0.6(2046.67)	21.69	98.74

Table 3: Comparison of move times to the specified solution quality

N_{size}	$Q(F)$	IGKLS	IKLS
		Move times	Move times
16	1.0(19.00)	402.2	450.9
64	0.8(95.00)	1932.3	2776.8
256	0.7(438.57)	13406.1	31173.3
1024	0.6(2046.67)	87931.8	1025314.0

times (move times) to reach the specified solution quality. A rough comparison can be made for the algorithm efficiencies by this comparison. The same benchmark instances are used in the comparison. Table 2 shows the average running times for 20 instances to reach the specified solution quality. The specified solution quality is set to smaller values for larger problems because the larger size problems become more difficult to reach high quality solutions. The values of the specified solution quality are the same as used in [5], and the results of IKLS in table 2 and table 3 were cited from [5]. As shown in table 2, the average running times of IGKLS are much shorter than those of IKLS, despite using a slower CPU in our method. Table 3 shows the average number of 1-swap moves of IGKLS and IKLS when the solutions reached the specified quality. Move times represent the sum of KLS move times and greedy (or kick) move times. Since the computational time is almost consumed by the evaluation of the cost function, the number of 1-swap move times is a good indicator of the computational complexity and the efficiency. The numbers of 1-swap move times of IGKLS is clearly much smaller than those of IKLS for all problem sizes. Thus, IGKLS is more efficient than IKLS.

From the above comparisons, it can be concluded that IGKLS is superior to IKLS which is the best metaheuristic algorithm for the node placement problem.

5. CONCLUSIONS

In this paper, we have proposed an Iterated Greedy algorithm with variable K-swap Local Search (IGKLS) for the node placement problem in bidirectional Manhattan Street Networks (BMSN). IGKLS consists of two central phases, the destruction phase that randomly removes some nodes from the solution, and the construction phase that reassigns the previously removed nodes using a greedy heuristic.

In our experiments, we compared IGKLS with iterated k-swap local search (IKLS) which is the best metaheuristic algorithm. Experimental results showed that IGKLS outperforms IKLS. Especially IGKLS was more efficient than IKLS for the large size problem.

To evaluate the algorithm performances more fairly, we need to reimplement IKLS and perform all tests on the same machine.

6. REFERENCES

[1] C. Baransel, W. Dobosiewicz, and P. Gburzynski. Routing in multihop packet switching networks: Gb/s challenge. *IEEE Network*, pages 38–61, May/June 1995.

[2] M. M. Freire and H. J. A. da Silva. Performance comparison of wavelength routing optical networks with chordal ring and mesh-torus topologies. In *International Conference on Networking (ICN) 2001*, pages 358–367, 2001.

[3] L. Jacobs and M. Brusco. A local search heuristic for large set-covering problems. *Naval Research Logistics Quarterly*, 42(7):1129–1140, 1995.

[4] K. Katayama, A. Hamamoto, and H. Narihisa. An effective local search for the maximum clique problem. *Information Processing Letters*, 95(5):503–511, 2005.

[5] K. Katayama, H. Yamashita, and H. Narihisa. Variable depth search and iterated local search for the node placement problem in multihop wdm lightwave networks. In *the 2007 IEEE Congress on Evolutionary Computation (CEC-2007)*, pages 3508–3515, September 2007.

[6] M. Kato and Y. Oie. Reconfiguration algorithms based on metaheuristics for multihop wdm lightwave networks. In *IEEE International Conference on Communications (ICC) 2000*, pages 1638–1644, June 2000.

[7] T. Kitani, M. Yonedu, N. Funabiki, T. Nakanishi, K. Okayama, and T. Higashino. A two-stage hierarchical algorithm for wavelength assignment in wdm-based bidirectional manhattan street networks. In *the 11th IEEE International Conference on Networks*, pages 419–424, 2003.

[8] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[9] E. Marchiori and A. Steenbeek. An evolutionary algorithm for large set covering problems with applications to airline crew scheduling. In *EvoWorkshops 2000*, pages 367–381, 2000.

[10] M. Marsan, G. Alberengo, A. Francesea, E. Leonardi, and F. Neri. All-optical bidirectional manhattan networks. In *IEEE International Conference on Communications (ICC) '92*, pages 1461–1467, 1992.

[11] B. Mukherjee. Wdm-based local lightwave networks part i: single-hop systems. *IEEE Network*, pages 12–27, May 1992.

[12] B. Mukherjee. Wdm-based local lightwave networks part ii: multihop systems. *IEEE Network*, pages 22–32, July 1992.

[13] K. A. Murthy, Y. Li, and P. M. Pardalos. A local search algorithm for the quadratic assignment problem. *Informatica*, 3(4):524–538, 1992.

[14] R. Ruiz and T. Stutzle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2007.

[15] S. R. Tiourine, C. A. J. Hurkens, and J. K. Lenstra. Local search algorithms for the radio link frequency assignment problem. *Telecommunication Systems*, 13(2-4):293–314, 2000.