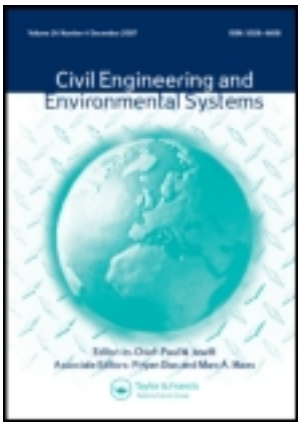


Civil Engineering and Environmental Systems
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcee20

Parallelised branch-and-bound algorithm for raster-based landfill siting

Kun-Hsing Liu [a] & Jehng-Jung Kao [a]
[a] Institute of Environmental Engineering, National Chiao Tung University , Hsinchu , Taiwan, Republic of China
Published online: 09 Aug 2012.

To cite this article: Kun-Hsing Liu & Jehng-Jung Kao (2013) Parallelised branch-and-bound algorithm for raster-based landfill siting, Civil Engineering and Environmental Systems, 30:1, 15-25, DOI: 10.1080/10286608.2012.709504

To link to this article: http://dx.doi.org/10.1080/10286608.2012.709504

Civil Engineering and Environmental Systems, 2013
Vol. 30, No. 1, 15–25, http://dx.doi.org/10.1080/10286608.2012.709504

Taylor & Francis
Taylor & Francis Group

Parallelised branch-and-bound algorithm for raster-based landfill siting

Kun-Hsing Liu and Jehng-Jung Kao*

Institute of Environmental Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

(*Received 27 February 2010*)

Landfill siting analysis is complex when compactness and other factors are simultaneously evaluated. In our previous work, a raster-based mixed-integer model was proposed to overcome this difficulty, and a C program was further developed to improve the computational time for solving the raster-based model. In this study, an enhanced parallelised branch-and-bound algorithm was proposed to shorten the solving time further. A parallelised computing environment with five computers was established for implementing the proposed algorithm. For comparison purpose, the un-parallelised algorithm was also tested on a single computer. The results show that the parallelised algorithm and computing environment can increase the speed by about three to seven times, while compared to the original algorithm implemented on a single computer.

Keywords: landfill; site selection; optimisation model; parallel processing; environmental systems analysis

1. Introduction

The site selection for constructing a landfill is a complex process because it requires performing analyses for various factors to evaluate site suitability. As suggested by Zyma (1990), an appropriate landfill site should have minimum impact on environment, society, and economy, comply with regulations, and receive general public acceptance. Implementing such a complicated procedure in a conventional information processing approach would be expensive and tedious. A geographic information system (GIS) is capable of processing a large amount of spatial data. Lindquist (1991) stated that using a GIS for landfill site selection increases objectivity and flexibility. Easy presentations to visualise GIS siting results are among its advantages also (Yesilnacar and Cetin 2005, Sener *et al.* 2006, Nas *et al.* 2010).

However, it is complex for landfill siting when compactness and other factors are simultaneously evaluated. Compactness represents the nature of the site and the extent to which it can be regarded as integrated tightly. The lower the level of compactness, the less likely the solution is to satisfy siting requirements, subsequently making general land planning difficult. Compactness can be defined by a variety of methods. For example, Wright *et al.* (1983) used the ratio of the perimeter to the area of a site as a measure of compactness. According to this definition,

*Corresponding author. Email: jjkao@mail.nctu.edu.tw

the shorter the perimeter of a site implies the higher its degree of compactness. Diamond and Wright (1989) applied the ratio of the largest diameter square to the area of a selected site as another measure. The largest diameter refers to the longest distance between any two points within the selected site. However, this method of calculation is nonlinear. Minor and Jacobs (1994) and Benabdallah and Wright (1992) adopted the former definition for a waste landfill siting problem and a land allocation problem, respectively. These spatially compactness models, although useful in solving a siting problem, have not been integrated into a GIS. Diamond and Wright (1989) indicated that such integration would provide an intelligent decision-making tool for land-use problems. The major obstacle to this integration is that significant numbers of integer variables and constraints are required to construct a compactness model for raster-based GIS map layers, thereby making the model difficult to solve by a general mixed-integer programming package.

In one of our previous studies (Kao and Lin 1996), we proposed an improved compactness model for raster-based GIS data. In that work, the model was applied to a case in central Taiwan. The model, although it uses less variables and constraints than two other models, still requires excessive computational time for a large raster-based landfill siting problem. Two primary reasons exist for this computational problem: (1) a large number of steps are implemented to search for a feasible integer solution during a typical branch-and-bound (B&B) solution procedure for solving a mixed-integer programming model; and (2) unnecessary branches are implemented on cells that are not contiguous. However, once a set of land cells is selected, the corresponding feasible integer solution can be easily determined without using a complex programming method. Moreover, branching on obviously impossible cells that are far away from previously selected cells is unnecessary. In our other previous study (Kao 1996), a C program was therefore developed, based on a proposed raster-based B&B algorithm, to implement multi-factor analyses for compactness and other siting factors with weights pre-specified by the user. The C program can avoid the two computational problems and significantly reduces the time required for solving the compactness model. The C program, although efficient in solving the compactness model, may still take a significant amount of time for a large problem. A parallelised algorithm was thus developed in this study to enhance further the performance of the C program for solving the compactness model.

The idea of a parallelised algorithm is to separate the computing being executed simultaneously by multiple processes or a cluster of computers. For example, Keedwell and Khu (2006) applied parallel computing to the multi-objective optimisation of two water distribution networks, and significant computational savings were observed. In this study, the previously developed B&B algorithm is modified for being able to implement in parallel without using a specific parallelised language. A parallelised computing environment with five single instruction single data computers has been established by two Perl (Wall *et al.* 2000) programs written by the authors. The developed parallelised algorithm and computing environment is applied to an illustrative example, without using any specific computer and language compiler.

2. The compactness model

The compactness model used in this study is adopted from our previous research. The development of the model and detail description and comparison to other models are referred to Kao and Lin (1996). The compactness model used in this work is listed as follows.

$$\text{Min} \sum_{i=0}^{i=m} \sum_{j=1}^{j=n+1} V_{i,j} \tag{1}$$

s.t.

$$2I_{i,j} - I_{i,j-1} - I_{i+1,j} + V_{i,j} \geq 0 \qquad \forall\, i \in \{0,\ldots,m\};\ \forall\, j \in \{1,\ldots,n+1\},$$

$$\sum_{i=1}^{i=m}\sum_{j=1}^{j=n} I_{i,j} \geq A \qquad \forall\, i \in \{1,\ldots,m\};\ \forall j \in \{1,\ldots,n\},$$

$$\sum_{i=1}^{i=m}\sum_{j=1}^{j=n} C_{i,j}^{k} \cdot I_{i,j} \geq G^{k} \qquad \forall k \in \{1,\ldots,p\},$$

$I_{i,j}$is [0,1] integer,

other constrains or bounds.

where m and n are the number of columns and rows of cells that represent the whole siting area; $V_{i,j}$ is used to record the length of the site perimeter; $I_{i,j}$, an [0,1] indicator variable, is defined to represent whether cell i, j belongs to a considered site; A is the required size (in numbers of cells) of the desired site; $C_{i,j}^{k}$ is the value of siting factor k for cell i, j; p is the number of considered factors; and G^{k} is the lower bound of the sum of factor values of cells in a site for siting factor k. Noticeably, to ensure that each cell in the siting area has an adjacent cell, a pseudo column of cells (for $j = n + 1$) on the right side and a pseudo row (for $i = 0$) of cells on the top side of the siting area are required and added. The continuity of the selected cells of the solution to the above model is guaranteed because the model seeks the smallest perimeter.

2.1. *The model with multiple factors*

For a problem with consideration of multiple factors, the objective function of the model should be modified as follows.

$$\text{Min} \sum_{i=0}^{i=m}\sum_{j=1}^{j=n+1}\left(w_v V_{i,j} + \sum_{k=1}^{p} w_k C_{i,j}^{k}\right), \tag{2}$$

where w_v is the weight for compactness and w_k is the weight for factor k. This modified objective function is based on the weighting method described by Cohon (1978). Solving this siting compactness model using a general optimisation package is time-consuming. Therefore, a special depth-first B&B algorithm (Kao 1996) was developed for solving the siting model efficiently.

3. The parallelised B&B algorithm

The special depth-first B&B algorithm is modified further in this study to parallelise the algorithm. The goal is to shorten the computational time for solving the siting problem. The concept of 'parallelized' is defined as a process being separated into several parts to obtain better computational efficiency, and each computer is assigned a part or more to solve.

3.1. *The procedure for implementing the parallelised B&B algorithm*

Figure 1 illustrates the conceptual procedure for implementing the parallelised algorithm. And the detail steps are described by the pseudo code listed below.

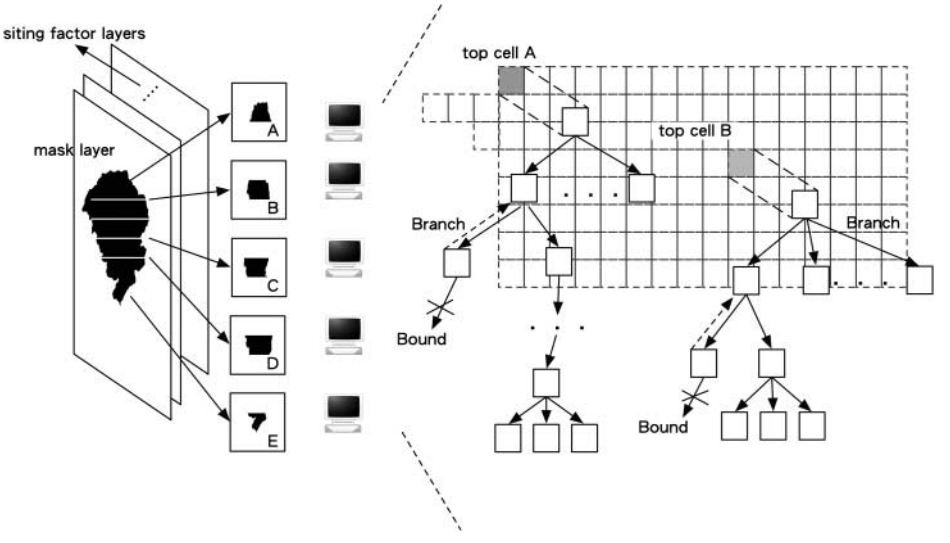


Figure 1. Parallelised B&B siting algorithm.

Algorithm: Parallel Siting

Read mask and factor map layers;
Read the options selected by the user and each computer is assigned to solve a pre-specified range of cells as top cells;
Activate each computer running in parallel to implement the *Algorithm: B&B Siting* for solving the siting problem for the pre-specified range of cells;
Display the final globally best result as the solution.

Algorithm: B&B Siting

For each candidate cell being severed as a top cell, do *Branch()*;

Branch():

Do Bound();
If a possible site is found, do *Check()* for checking feasibility and/or noninferiority;
Collect candidate cells that can be branched into an adjacent cell stack;
For each cell in the adjacent cell stack, do *Branch()*;

Bound():

Implement bounding rules to prune subtrees that are unnecessary to branch further: possible cells to branch, bounds of factor values and siting area, bounds of estimated site factor values, improvement of the objective function value based on an estimated current objective value, maximally allowable width and/or height to the top cell, width of a horizontal bridge, and number of corner cells.

Check():

Check feasibility with constraints provided by the user;

Check noninferiority; (optional)

If it is a valid site, output or record the associated information.
If the objective value of the valid site is better than the locally best one, then the locally best one is updated.
If the locally best one is better than the globally one, the global one is replaced by the local one; otherwise the global one is copied to the local one.

3.2. *The branch step*

The detailed B&B steps of this parallelised algorithm are similar to those developed by Kao (1996). A brief description of both steps is provided below and details of both steps are referred to Kao (1996). The branch step starts with a top cell from a siting area. Next, all candidate branching cells, cells not eliminated after applying bounding rules described later for the *Bound* step of this algorithm, of the top cell are added into a branching pool for current searching level. One of the candidate branching cells is then selected (branched) to implement the next level of searching. The candidate branching cells for this newly branched cell are collected into the branching pool for next searching level. This procedure is repeated until the number of selected cells satisfies the required siting size. Such a set of selected cells forms a site and is passed to the *Check* step to check for its feasibility, noninferiority (optional), and validity. If the site passes all the checks, its objective and factor values are recorded, and the best values obtained so far are used as bounds. Then, another site is formed by replacing the most recently selected cell by another cell in the branching pool. This procedure is repeated until no new cell in the branching pool in current level. The searching process then moves up one level in the branching tree and continues the searching from another candidate branching cell in the branching pool of the upper level. If no new cell can be branched, the searching process moves further up in the tree. This procedure is repeated to ensure that all branching pools are empty and each cell has been used as a top cell.

3.3. *The bound step*

This step is applied to prune subtrees of the depth-first branching tree that are not necessary to explore further. Pruning the subtrees as early as possible would significantly save computational time. Several bounding rules are provided: possible cells to branch, bounds of factor values and siting area, bounds of estimated site factor values, improvement of the objective function value based on an estimated current objective value, maximally allowable width and/or height to the top cell, width of a horizontal bridge, and number of corner cells. The required size of a site is known, thereby making it unnecessary to branch on cells that are too far away from the current top cell. The continuity of the finally selected site is guaranteed because a site with unconnected cells has a poor compactness value and will not be selected.

The estimated value of a factor for a site is computed by the following equation.

$$F_e = \sum_{i=1}^{s} f_i + \sum_{j=s+1}^{r} f_j^l, \tag{3}$$

where F_e is the estimated value of a factor for the currently selected site; s is the number of currently selected cells; f_i is the factor value of cell i; r is the number of cells required for a site; and f_j^l is the lower bound of the factor value of cell j. If the estimated objective function value is worse than the best one currently recorded so far, the subtree following the current cell can be pruned. The user can also define the maximally allowable width and/or height to the top cell, the width of a horizontal bridge, or the number of corner cells. The default values for these rules avoid branching on those cells that are too far away from previously selected cells and searching for sites having a poor shape. A corner cell is a cell without any cell that is selected on its left-hand and top sides. If the number of corner cells is limited to 1, all sites selected tend to have rectangular shapes.

Properly using these rules can significantly reduce the size of the branching tree. Cells that are not eliminated in this *Bound* step are candidate branching cells. If any candidate branching cell exists, the *Branch* step is repeated to branch from one of candidate branching cells. The two steps are repeated until no candidate branching cell is available on the branching tree.

3.4. *The parallelised B&B step*

The entire siting area is divided into several smaller parts and the B&B algorithm is implemented by several computers simultaneously in parallel. The computing speed could be enhanced because the branching tree for each part is not so large as for the entire siting area. Although multiple parts could be solved as multi-processes on the same one-CPU computer, this situation is not truly parallelised. These parts are more efficient to be solved in parallel by several computers simultaneously. Whether being solved on single or multiple computers, the current best result during the B&B searching processes is shared by all processes.

The parallelised B&B algorithm is implemented by a C program and two Perl (Wall *et al.* 2000) computer programs written by the authors. The Perl programs are described in the next section. The C program is modified from the program developed by Kao (1996). The program first read a mask map layer, indicating valid siting areas and cells, and several factor map layers, providing cell suitability scores of siting factors that used to compute objective function values during the B&B searching process. Several options can be set by the user. Available major options include minimal and maximal size limits of the site to be searched, the best objective value currently known or a best guess, an option to check the noninferiority of a site, maximally allowable horizontal bridge width of a site, weights of siting factors considered in the objective function, maximally allowable width and height to the top cell, and other options.

Multiple computers are run in parallel and each one implements the B&B algorithm for a separate range of land cells specified by the user. During the B&B searching process, if any superior solution is obtained, the bounding record for the current best one is replaced by the new one.

4. Parallel computing environment

Five typical personal computers, one server and five clients, were used to establish a parallel computing environment. We did not use the popular Cluster environment (e.g. MPICH2 cluster: Gropp *et al.* 2009) because it is too complex and unaffordable for a local environmental engineering office. Instead, this study developed two Perl (Wall *et al.* 2000) programs to establish a parallel computing environment for implementing the proposed parallelised B&B algorithm. The programs implement a client–server environment, one for the server side and the other for client sides. Immediately after the Perl program on the sever computer is executed, it initiates the programs on all client computers too. The Perl programs apply a socket interface over the computer network to achieve the initiation. The program provides a function for network data flow control and two-way transmission of data. The socket interface accepts request and issue new commands over the network between the server and client computers. The bounding record for currently best one is stored on one of the computers and shared to others via a network file sharing protocol. A read/write locking mechanism is applied to read and update the bounding record. Whenever a new feasible siting solution is obtained from any computer, the program will check whether the new solution is better than the currently recorded best solution. If the new solution is worse, then it is discarded; if it is superior, then a write lock is issued to update the sharing record for storing the new one as the currently recorded best bound. As shown in Figure 2, during the parallel execution, the globally best result is shared to all computers. Since file writing and reading over the network is time consuming, a local copy of the globally best result is made for each client computer as the locally best result. Whenever a new local result is superior to the locally best one, it becomes the new locally best result. At the same time, the server-side checks whether the locally best result is better than the global one or not. If the local one is better than the global one, the global one is replaced with the new one.

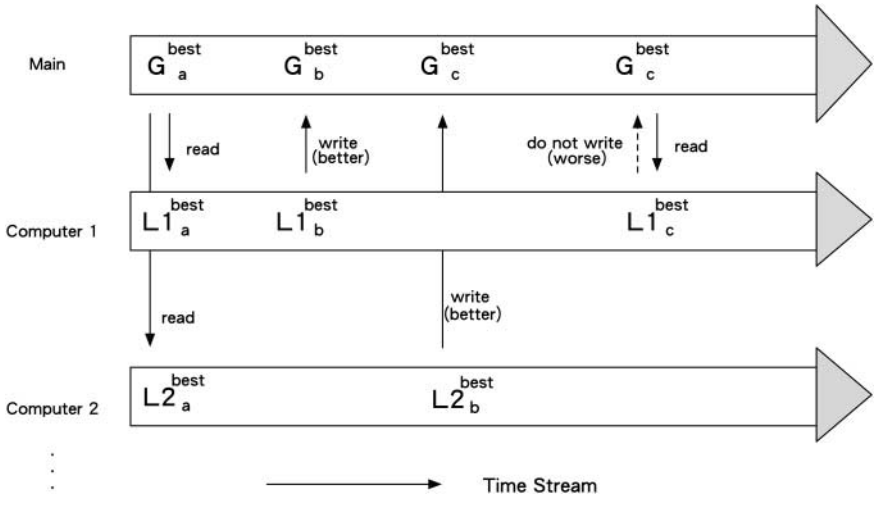


Figure 2. Read/write mechanism for reading and updating currently recorded best bound.

All computers are with Intel^{TM} Pentium CPU and each costs less than US$800. All computers are installed with the famous Linux (Fedora project 2009) operation system. The server computer is not only a server for controlling the entire parallel siting solving process, but also used as a client to participate in solving a siting problem. All client computers, each implements a B&B siting C program, are run in parallel. This computing environment is cheap and effective for implementing the proposed parallelised siting model.

5. Illustrative case

An illustrative case for a landfill siting problem in central Taiwan was applied to demonstrate the applicability of the proposed parallelised B&B siting algorithm. The mask map layer consists of 2500 land cells. Three major siting factors are considered: land slope, land soil, and land cost. Figure 3 shows factor map layers prepared for the study area, for which the siting area is divided into numerous square land cells. The factor map layers are created and digitised using a scoring system that is based on the soil type, slope, and cost of each land cell. For instance, a soil type with lower infiltration rate is assigned a lower suitability score, and vice versa. In this case, for consistency with the compactness model minimising its objective function, a low value implies a high suitability and a high value implies a low suitability. A map overlay function is applied to determine the final score of each land cell based on these factor suitability maps. The data for all map layers are shared to the server and client computers in the established parallel computing environment.

As shown in Figure 1, each computer implements the parallelised B&B program by starting from a different top cell in the siting area. Four scenarios were analysed: in the first three scenarios, each siting factor is considered independently, and in the last one, all three factors are considered simultaneously. The scenarios are (A) land slope; (B) land soil; (C) land cost; and (D) all three factors. All of the values for each factor layer are normalised to the range from 1 to 100, where 1 is the best. For comparison purpose, the siting problem was also solved by a single computer to execute the B&B siting algorithm. Figure 4 presents the solutions obtained in each scenario for a size limit of 16 land cells. The weight set used in each scenario is also listed in the same figure. Figure 5(a) compares the computational performance for four scenarios with a

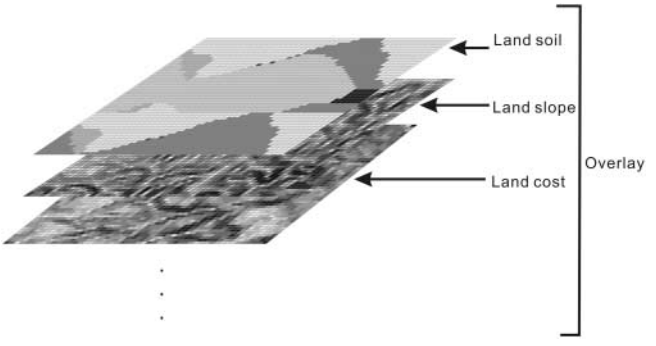
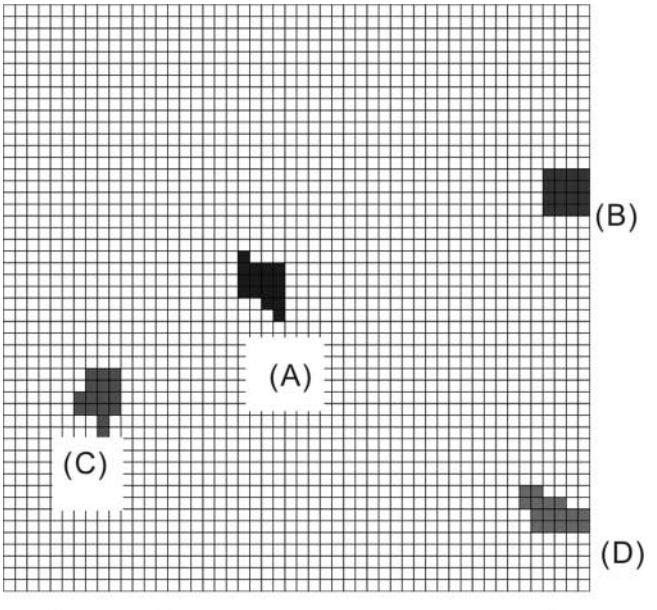


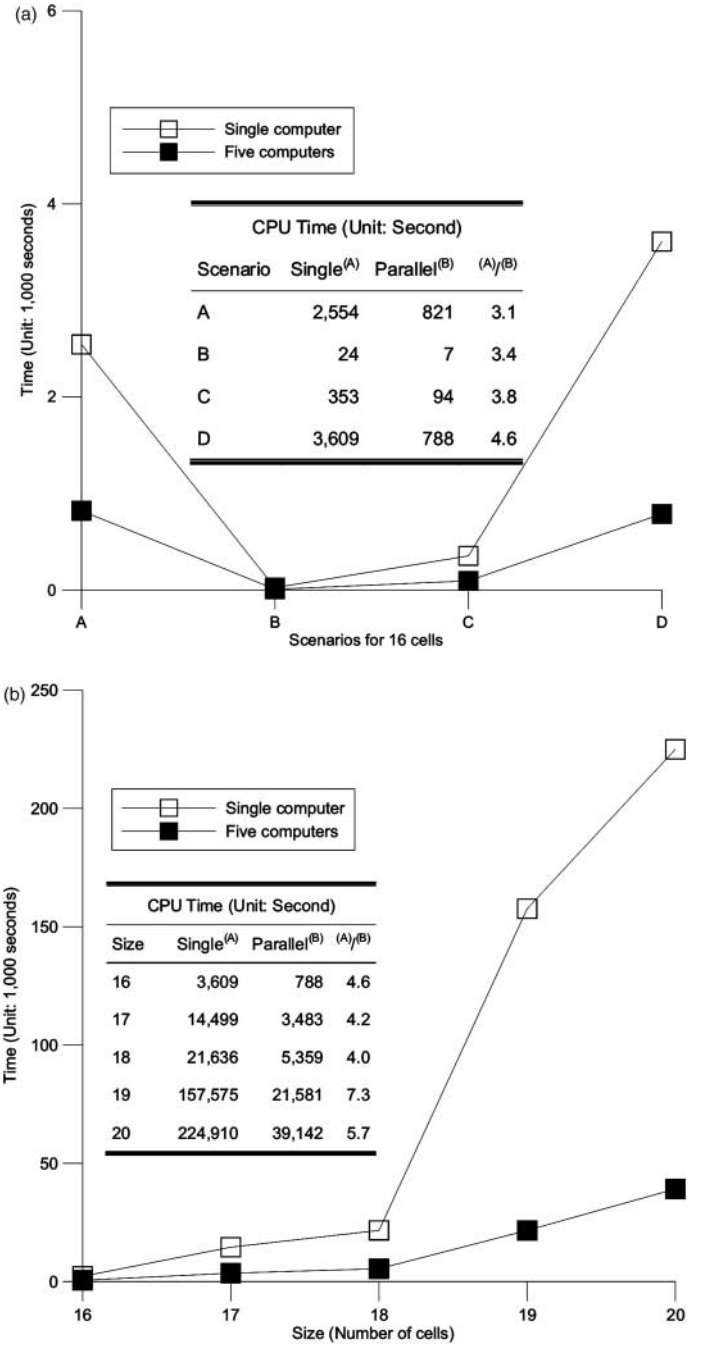
Figure 3. Suitability value maps for landfill siting factors.



Scenario	Weight		
	Land slope	Land soil	Land cost
A	1	0	0
B	0	1	0
C	0	0	1
D	1	1	1

Figure 4. Results obtained in different scenarios with a 16-cell size limit.

size limit of 16 land cells. The computational time required by the parallel model is 1/3.1–1/4.6 times that required when a single computer is used. The computational time required for scenarios that involve only one factor is short, even when a single computer is used. However, the problem becomes more complicated when all three factors are simultaneously considered and a much longer computational time is also required for solving the multiple-factor siting problem. Figure 4 shows the locations of solutions obtained for different scenarios. The computational time is substantially shorter than that of a single computer, as presented in Figure 5(a). The typical reduction is expected to be 1/5 because five computers were used. All the cases did not achieve



CPU Time (Unit: Second)

Scenario	Single(A)	Parallel(B)	(A)/(B)
A	2,554	821	3.1
B	24	7	3.4
C	353	94	3.8
D	3,609	788	4.6

CPU Time (Unit: Second)

Size	Single(A)	Parallel(B)	(A)/(B)
16	3,609	788	4.6
17	14,499	3,483	4.2
18	21,636	5,359	4.0
19	157,575	21,581	7.3
20	224,910	39,142	5.7

Figure 5. CPU time required to solve the siting model on a single computer and on five parallel computers (a) four scenarios with a 16-cell size limit; (b) scenario D with size limits of 16–20 cells.

this typical reduction because (1) extra time is required for I/O (read/write); (2) work balance was roughly done by assigning approximately the same range of cells and it might not be the true work balance among computers. However, the computational time reduction is already quite significant and is enough to demonstrate the success of the proposed algorithm.

Various size limits of 16–20 land cells are also tested for Scenario D and their computational times are compared. All factors are assigned equal weight in the siting objective function. Figure 5(b) presents the results obtained for various size limits using both a single computer and five computers in parallel. The computational time required to solve the siting problem using the parallelised algorithm is significantly reduced to 1/4–1/7.3 times that required by a single computer. As for a size limit of 16–20 cells, three of the cases did not exhibit the typical computational time reduction of 1/5. However, for size limits of 19 and 20 cells, further time reductions of 1/7.3 and 1/5.7, respectively, were achieved because (a) the I/O time is less influential while compared to the long computational time that is required to solve the problem with a large size limit; and (b) a good solution is obtained by one of the five computers in the early stage of the B&B searching procedure and numerous unnecessary branches are pruned, significantly reducing the computational time.

6. Conclusion

The proposed parallelised B&B algorithm substantially shortens the computational time for solving a siting problem. The concept of this new algorithm is to divide the process of a siting problem into several small processes, and each computer executes a small process. For implementing the parallel B&B algorithm, this research establishes a parallel computing environment. The computing environment consists of five cheap personal computers, which are affordable by a local engineering office. When the parallel B&B algorithm is applied to the landfill siting problem in central Taiwan, the computational time is significantly reduced below that required using a single computer. Furthermore, the proposed algorithm and computing environment make it easily to integrate them with a typical geographical information system.

Acknowledgement

The authors thank the National Science Council of the Republic of China for providing partial financial support of this study under Grant No. NSC 95-2221-E-009-115.

References

Benabdallah, S. and Wright, J.R., 1992. Multiple subregion allocation models. *Journal of Urban Planning and Development*, 118 (1), 24–40.

Cohon, J.L., 1978. *Multiobjective programming and planning*. New York: Journal of Academic Press.

Diamond, J.T. and Wright, J.R., 1989. Efficient land allocation. *Journal of Urban Planning and Development*, 115 (2), 81–96.

Fedora Project, 2009. *Fedora core linux* [online]. Fedora project. Available from: http://fedora.redhat.com [Accessed 30 April 2009].

Gropp, W., Lusk, E., Ashton, D., Balaji, P., Buntinas, R., Chan, A., Goodell, D., Mercier, G., Ross, R., Thakur, R., and Toonen, B., 2009. *MPICH2 user's guide* [online]. Argonne National Laboratory Group, Mathematics and Computer Science Division Argonne National Laboratory, Argonne IL. Available from: http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-1.0.8-userguide.pdf [Accessed 30 April 2009].

Kao, J.J., 1996. A raster-based C program for siting a landfill with optimal compactness. *Computers and Geoscience*, 22 (8), 837–847.

Kao, J.J. and Lin, H.Y., 1996. Multifactor spatial analysis for landfill siting. *Journal of Environmental Engineering*, 10 (4), 307–317.

Keedwell, E. and Kuh, S.-H., 2006. A novel evolutionary meta-heuristic for the multi-objective optimization of real-world water distribution networks. *Engineering Optimization*, 38 (3), 319–336.

Lindquist, R.C., 1991. Illinois cleans up: using GIS for landfill siting. *Geographic Information Systems*, February, 30–35.

Minor, S.F. and Jacobs, T.L., 1994. Optimal land allocation for solid- and hazardous-waste landfill siting. *Journal of Environmental Engineering*, 120 (5), 1095–1108.

Nas, B., Cay, T., Iscan, F., and Berktay, A., 2010. Selection of MSW landfill site for Konya, Turkey using GIS and multi-criteria evaluation. *Environmental Monitoring and Assessment*, 160 (1), 491–500.
Sener, B., Süzen, M.L., and Doyuran, V., 2006. Landfill site selection by using geographic information systems. *Environmental Geology*, 49 (3), 376–388.
Wall, L., Christiansen, T., and Orwant, J., 2000. *Programming Perl.* 3rd ed. Sebastopol, CA: O'Reilly & Associates, Inc.
Wright, J., Revelle, C., and Cohon, J., 1983. A multiobjective integer programming model for the land acquisition problem. *Regional Science and Urban Economics*, 13 (1), 31–53.
Yesilnacar, M.I. and Cetin, H., 2005. Site selection for hazardous wastes: A case study from the GAP area, Turkey. *Engineering Geology*, 81 (4), 371–388.
Zyma, R., 1990. Siting consideration for resource recovery facilities. *Public Works*, 121 (10), 84–86.