Int j simul model 11 (2012) 3, 150-164
ISSN 1726-4529 Original scientific paper

SIMSUM1: A GENERAL OPTIMISATION VIA SIMULATION APPROACH FOR 0-1 PROGRAMMING MODELS

Azimi, P.
Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University,
Argentina Sq., Ahmad-e-Qasir St., 13th Aven., No. 1, Flat 9, Tehran, Iran (P. O. Box: 1513815531)
E-Mail: p.azimi@yahoo.com

Abstract
The current study develops a new general heuristic approach to address a special class of combinatorial problems, efficiently. The approach combines discrete event simulation together with relaxation techniques to solve a special class of 0-1 programming models including a constraint that restricts the summation of all variables to be 1. Three well-known combinatorial problems -including such a constraint- such as Dynamic Facility Layout Problem (DFLP), Graph Labelling Problem (GLP) and Travelling Salesman Problem (TSP) have been addressed and could be solved efficiently by the proposed algorithm. Several experiments have been carried out to show the efficiency of the algorithm. The results show that the proposed algorithm can be used for several real world applications while according to the best knowledge of the author, at least for DFLP and GLP, it is the fastest algorithm which has been developed in the literature.
(Received in October 2011, accepted in April 2012. This paper was with the author 2 months for 3 revisions.)

Key Words: Discrete Event Simulation, Dynamic Facility Layout Problem, Graph Labelling Problems, Travelling Salesman Problem, Zero-One Programming

1. INTRODUCTION

In the zoo of combinatorial problems, the majority of problems are referred to NP-hard or NP-complete according to their complicated structures. Unfortunately, the case is coming through when one finds out that nearly all applicable problems which could be used in real world environments, have such a complexity structure. Therefore, development of the efficient heuristic algorithms has been absorbed a lot of attentions among the researchers. However, it is out of the current research targets to address all combinatorial problems, but three famous examples have been introduced. All examples have two main similarities:

- They are 0-1 programming models.
- They have a special constraint which restricts the summation of all decision variables to be 1.

The two important similarities are the main keys causing the proposed algorithm to be the most efficient one in the combinatorial optimisation literature with the mentioned characteristics. However, in section 2, the reason above separating this class of combinatorial optimisation problems will be explained.

1.1 Facility layout problems

According to [1], the problem associated with the placement of facilities in a plant area, often referred to as the "facility layout problem" is known to have a significant impact on manufacturing costs, work in processes, lead times, and company productivity. According to [2-4], a well-designed placement of facilities contributes the overall efficiency of operations and can reduce up to 50 % of the total operating expenses. A facility may be a manufacturing

plant, warehouse, port, administrative office building or a service centre. Nowadays, manufacturing plants must be able to respond quickly to changes in demand, production volume and product mix. [5] reported that, on average, 40 % of company's sales come from new products. However, changes in product mixes would require modifications to the production flow, thus affecting the facility layout. Recently, the concept of dynamic layout problems has been introduced by several researchers. According to [6-7], dynamic layout problems take into account possible changes in the material handling flow over multiple periods. In the dynamic facility layout problem (DFLP), the planning horizon is generally divided into certain periods which may be defined in weeks, months, or even years. The flow data for each period are forecasted, and it is assumed that these data remain constant throughout the defined period. [8] was the first researcher who developed an optimisation approach based on a dynamic programming model for the DFLP. The author showed that a number of layouts had to be evaluated to guarantee optimality for a DFLP with *N* departments and *T* periods is $(N!)^T$. It should be mentioned that in addition to exact algorithms, many meta-heuristic algorithms have been reported in the literature. [9] developed a genetic algorithm to solve the DFLP, and [10] used a tabu search (TS) heuristic. Their TS heuristic is a two-stage search process that incorporates diversification and intensification strategies. [11] presented a simulated annealing (SA) heuristic for the DFLP. [12] presented a hybrid genetic algorithm, and [13] proposed a new heuristic to solve the DFLP. [14] developed three hybrid ant systems (HAS). In addition, [15] presented two (SA) heuristics. The first, (SA I), is a direct adaptation of SA for the DFLP. The second, (SA II), is the same as SA I, except the fact that it incorporates an added look-ahead/look-back strategy. For an extensive review on the DFLP, one can refer to the work of [16].

In the class of Facility Layout Problem, we consider Dynamic facility layout problem with budget constraints, because the structure is much more complicated and real than static one without the company budget for allocating of facilities. There are few studies on DFLP with the budget constraints ([8], [17], [2], and [18]). [17] developed a simulated annealing algorithm for the problem. According to the latest study, [18] used simulation to address a DFLP with budget constraint. The DFLP can be modelled as a modified quadratic assignment problem, similar to the static facility layout problem (SFLP). The notations used in the model are given below:

$$X_{tij} = \begin{cases} 1 & \text{if department } i \text{ is assigned to location } j \text{ at period } t, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where *N*: both the number of departments and the number of locations, *T*: the number of periods in the planning horizon, C_{tijkl}: cost of material handling between department *i* in location *j* and department *k* in location *l* during period *t*, A_{tijl}: cost of rearranging department *i* from location *j* to location *l* at the beginning of period *t*, LB_t: left-over budget from period *t* to period *t* + 1, B_t: available budget for period *t*, AB_t: allocated budget for period *t*.

Problem 1:

$$Min.Z = \left(\sum_{t=2}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{l=1}^{N} A_{tijl} * X_{t-1,ij} * X_{til} + \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} C_{tijkl} * X_{tij} * X_{tkl} \right) \quad (2)$$

s.t.

$$\sum_{j=1}^{N} X_{tij} = 1, \ \forall i = 1, 2, \dots, N, \ \forall t = 1, 2, \dots, T \quad (3)$$

$$\sum_{i=1}^{N} X_{tij} = 1, \ \forall j = 1, 2, \dots, N, \ \forall t = 1, 2, \dots, T \quad (4)$$

$$LB_t = B_t - \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{l=1}^{N} A_{tijl} * X_{t-1,ij} * X_{til}, \ \forall t = 1, 2, \dots, T \quad (5)$$

$$B_t = AB_t + LB_{t-1},\ \forall t = 1, 2, \dots, T \quad (6)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl} * X_{t-1,ij} * X_{til} \le B_t,\ \forall t = 1, 2, \dots, T \quad (7)$$

$$X_{tij} \varepsilon \{0,1\},\ \forall i, j = 1, 2, \dots, N,\ \forall t = 1, 2, \dots, T \quad (8)$$

$$LB_t, B_t, AB_t \ge 0,\ \forall t = 1, 2, \dots, T \quad (9)$$

In Problem 1, the objective function (2) is used to minimize the sum of the rearrangement and material handling costs. Constraint set (3) restricts each location to be assigned to only one department during each period and constraint set (4) ensures that exactly one department is assigned to each location within each period. Constraint set (5) is for equating the total available budget in a period to sum of the leftover budget from previous period and the allocated budget in the current period. Finally, the constraint set (6) represents the budget constraints for each period. According to [2], this zero-one programming problem has been shown to be a NP-hard model. According to problem 1, all variables have zero or one values and the summation of all variables is 1 (constraint set 3 or 4). In fact, the facility layout problem is a class of combinatorial problems that is suitable for the proposed algorithm. Problem 1 can be changed to a relaxed 0-1 programming model (Problem 2) which is a linear programming model with continuous variables by using two sets of new variables as follows:

$$X_{t-1,ij} + X_{til} = 2\, Z_{t-1,tijl} \quad (10)$$

$$X_{tij} + X_{tkl} = 2\, W_{tijkl} \quad (11)$$

Problem 2:

$$Min.Z = \left(\sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl} * Z_{t-1,tijl} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} C_{tijkl} * W_{tijkl} \right) \quad (12)$$

s.t.

$$\sum_{j=1}^{N} X_{tij} = 1,\ \forall i = 1, 2, \dots, N,\ \forall t = 1, 2, \dots, T \quad (13)$$

$$\sum_{i=1}^{N} X_{tij} = 1,\ \forall j = 1, 2, \dots, N,\ \forall t = 1, 2, \dots, T \quad (14)$$

$$X_{t-1,ij} + X_{til} = 2 * Z_{t-1,tijl},\ \forall i, j, l = 1, 2, \dots, N,\ \forall t = 2, 3, \dots, T \quad (15)$$

$$X_{tij} + X_{tkl} = 2 * W_{tijkl},\ \forall i, j, k, l = 1, 2, \dots, N,\ \forall t = 1, 2, \dots, T,\ i \ne k, j \ne l \quad (16)$$

$$LB_t = B_t - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl} * Z_{t-1,tijl},\ \forall t = 1, 2, \dots, T \quad (17)$$

$$B_t = AB_t + LB_{t-1},\ \forall t = 1, 2, \dots, T \quad (18)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl} * Z_{t-1,tijl} \le B_t,\ \forall t = 1, 2, \dots, T \quad (19)$$

$$0 \le X_{tij}, \le 1,\ \forall i, j = 1, 2, \dots, N,\ \forall t = 1, 2, \dots, T \quad (20)$$

$$0 \le Z_{t-1,tijl} \le 1, \forall i, j, l = 1, 2, \dots, N,\ \forall t = 2, 3, \dots, T \quad (21)$$

$$0 \le W_{tijkl} \le 1,\ \forall i, j, k, l = 1, 2, \dots, N,\ \forall t = 1, 2, \dots, T,\ i \ne k,\ j \ne l \quad (22)$$

$$LB_t, B_t, AB_t \ge 0,\ \forall t = 1, 2, \dots, T \quad (23)$$

In section 2, the proposed approach will be introduced in details. Then, DFLP will be solved by this algorithm.

1.2 Travelling salesman problem

Travelling salesman problem (TSP) is one of the most challenging problems among combinatorial optimisation methods. Its importance stems from the fact that there is a plethora

of fields in which it finds applications. Among all applications of this problem we can address some vast useful applications like optimizing the AGV (Automatic guided vehicles) routes in a production system and transportation problems like post office delivery one or in a large warehouse transportation system. To review all details regarding the applications of TSP one can refer to [19]. Because the TSP has been proved to belong to the class of NP-hard problems [20], heuristics and meta-heuristics have a majority in the developed solution methods. Today, most researchers try to find better heuristics for solving a wide range of large-scale instances of the TSP. [21] used and improved genetic algorithm with reinforcement mutation called RMGA to solve the TSP and showed that the result quality of their algorithm is better than previous works. [22] used a branch and bound algorithm together with simulation technique to solve several TSPs. They showed that their heuristic algorithm had been the most efficient algorithm developed so far. To formulate the asymmetric TSP on *m* nodes:

$$x_{ij} = \begin{cases} 1 & \text{if } i \to j \quad \text{is in the tour} \\ 0 & \text{otherwise} \end{cases} \quad i \neq j, i, j = 1, 2, \dots, m \tag{24}$$

Problem 3:

$$Min.Z = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij} \tag{25}$$

s.t.

$$\sum_{j=1}^{m} x_{ij} = 1, \ \forall i = 1, 2, \dots, m \tag{26}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \ \forall j = 1, 2, \dots, m \tag{27}$$

$$\sum_{i \in A} \sum_{j \in A} x_{ij} \leq |A| - 1, \ \forall A \subset \{1, 2, \dots, m\} \tag{28}$$

$$x_{ij} = \{0,1\}, \ \forall i, j = 1, 2, \dots, m \tag{29}$$

where A is any nonempty proper subset of the nodes 1,..., m. The cost c_{ij} is allowed to be different from the cost c_{ji}. According to Problem 3, the TSP is a 0-1 programming model that includes special constraints (26) and (27) which make it suitable for the proposed heuristic algorithm.

1.3 Labelling problems

All graph labelling methods trace their origin to the one which was introduced by [23] or the one given by [24]. A labelling (or valuation) of a graph G is a map that carries graph elements to some special numbers (usually to the positive or non-negative integers). If we review all studies from that time to the current time, we can find that major studies are focused on proving a special labelling for different class of graphs. For instance, we can refer to the one that was collected by [25] which covers all studies conducted in the field of graph labelling problems. According to [26], because the graph labelling problems have some applications in industries, therefore some researches focused on how to label a graph by using programming languages such as constraint programming or mathematical programming [27-30]. In these studies, the researchers intended to develop an algorithm to find a feasible solution of a special labelling for a given graph.

[23] called a function f, a β-valuation of a given graph $G = (V, E)$, if f is an injective from the V to the set $\{0, 1, \dots, \|V\|\}$ such that we assign label $\|f(u) - f(v)\|$ to the edge uv, the resulting edge labels are distinct. The most common choices of domain are the set of all vertices and edges (such a labelling is called total labelling), the vertex-set alone (vertex-labelling), or the edge-set alone (edge-labelling). Other domains are also possible. [31] called

such a labelling as graceful labelling. [25] categorized all labelling problems to 5 groups which called them: Graceful labelling, Harmonious labelling, Magic-type labelling, Antimagic-type labelling and Miscellaneous labelling. [32] introduced the notion of vertex-magic total labelling. For a Graph $G = (V, E)$ an injective mapping f from $V \cup E$ to the set $\{1, 2, \dots, |V| + |E|\}$ is a vertex-magic total labelling if there is a constant k, called the magic constant, such that for every vertex v, $f(v) + \sum f(vu) = k$, where the sum is over all vertices u adjacent to v, however some authors use the term "vertex-magic" for this concept. This property is called "magic sum". [33] showed several applications of graph labelling problems including communication networks, radar coding/decoding procedures and being suitable lower bounds for some so-called mathematical problems like TSP. According to [30], denote the vertices of the graph $G = (V, E)$ by $v_1, v_2, \dots, v_n$, which has n vertices and m edges. Now, define the decision variables of the model as follows:

x_i – the label of vertex v_i, $i = 1, 2, \dots, n$,
x_{ij} – the label of an edge (v_i, v_j), $i, j = 1, 2, \dots, n$, $i \neq j$,
k – the magic constant,
w_{it} – A, 0-1 variable, where $w_{it} = 1$ implies that $x_i = t$, $i = 1, 2, \dots, n$, $t = 1, 2, \dots, m+n$,
r_{ijt} – A, 0-1 variable where $r_{ijt} = 1$ implies that $x_{ij} = t$, $i, j = 1, 2, \dots, n$, $i \neq j$, $t = 1, 2, \dots, m+n$.

The following 0-1 model has a feasible solution,. if $G = (V, E)$ has vertex-magic total labelling.

Problem 4:

$$x_i + \sum_{(v_i,v_j)\in E(G)} x_{ij} = k, \ \forall i = 1, 2, \dots, n \tag{30}$$

$$x_i = \sum_{t=1}^{m+n} t w_{it}, \ \forall i = 1, 2, \dots, n \tag{31}$$

$$x_{ij} = \sum_{t=1}^{m+n} t r_{ijt}, \ \forall (v_i, v_j) \in E(G) \tag{32}$$

$$\sum_{t=1}^{m+n} w_{it} = 1, \ \forall i = 1, 2, \dots, n \tag{33}$$

$$\sum_{t=1}^{m+n} r_{ijt} = 1, \ \forall (v_i, v_j) \in E(G) \tag{34}$$

$$\sum_{i=1}^{n} w_{it} + \sum_{(v_i,v_j)\in E(G)} r_{ijt} = 1, \ \forall t = 1, 2, \dots, m+n \tag{35}$$

$$x_i, x_{ij}, k \geq 0, \ \forall i = 1, 2, \dots, n, \ \forall (v_i, v_j) \in E(G) \tag{36}$$

$$w_{it}, r_{ijt} = \{0,1\}, \ \forall i = 1, 2, \dots, n, \ \forall (v_i, v_j) \in E(G), \ \forall t = 1, 2, \dots, m+n \tag{37}$$

Problem 4 has $4n + 3m$ constraints and $(n+m)(n+m+1)+1$ variables. Note that in all labelling problems, there is no objective function because no optimisation occurred. We just search for a feasible solution in this kind of combinatorial optimisation problem. According to Problem 4, the Vertex-magic labelling problem is 0-1 programming model with special constraint sets (33) and (34). Therefore, the class of labelling problems are another class of suitable combinatorial optimisation problems for the proposed algorithm. According to constraint sets (31) and (32), x_i and x_{ij} are integer variables which are dependent on w_{it} and r_{ijt}, respectively and because w_{it} and r_{ijt} are 0-1 variables, therefore just w_{it} and r_{ijt} will be modelled in the simulation model.

2. PROPOSED ALGORITHM

The proposed algorithm was named as "SimSum1", because:

- The basic engine of the algorithm is discrete event simulation, and
- It solves a special class of combinatorial problems where they have 0-1 variables and the summation of all variables is equal to 1.

In order to have better view on the proposed algorithm, its structure for DFLP has been described here. However, for other combinatorial problems, the structure is straight forward. In Problem 1, the decision variable values (x_{tij}) could be just 0 or 1 and this is the main problem that makes solving process of this Problem very hard. In contrast, all decision variables in Problem 2 (the relaxed form of problem 1) are not integers and have continuous values. This fact makes this problem very easy to be solved but the optimal solutions of Problem 2 may not be feasible for Problem 1, because of their continuous values. In the best case, assume that all optimum solutions in Problem 2 have integer values. Therefore, the optimum solutions of Problem 1 have been found, but this is a rare case. Problem 1 must have a unimodularity property that when it is relaxed, the optimum solutions of resulting problem are integers. A DFLP is a NP-Hard problem, so it cannot be uinimodular, in general. Therefore, there may be some non-integers values among optimum solutions of Problem 2 which are infeasible for Problem 1. For example assume that in Problem 2, the optimum values of assigning facility *i* to some locations at period *t* (x_{tij}) are not integers. Because $x_{tij} \geq 0, \forall i, j, t$ and according to constraint set (13), for facility *i* at period *t*, the summation is equal to 1 over all locations, so ($x_{tij}, \forall j$) could be interpreted as a probability distribution function of assigning facility *i* to each location at period *t*. Because Problem 2 is the relax form of the main Problem (Problem 1) with the same objective function and constraints, the optimum solutions are assumed to be close enough to real optimum solutions of Problem 1. Therefore, SimSum1 uses these probability distribution functions for random assignments in the simulation model. Assume that we have 5 facilities, 5 locations and 5 periods and we have $x^*_{111} = 0.05, x^*_{112} = 0.08, x^*_{113} = 0.02, x^*_{114} = 0.00, x^*_{115} = 0.85$ in the optimum solution of Problem 2 which is a partial optimum solution for facility 1 at period 1. In SimSum1, the simulation model uses these optimum values to generate random assignments for facility 1 at period 1 to any locations, i.e., assigns facility 1 to location 1 with probability of 5 %, to location 2 with the probability of 2 %, ... in period 1. It means, for example after 100 replications and generating 100 random assignments of facilities to locations in every period, around 5 times, facility 1 has been assigned to location 1, around 2 times to location 2, ... in period 1. In fact, instead of using a blind random assignment like Uniform distribution (Fig. 1), the simulation model uses a special distribution function which comes from the optimal solutions of Problem 2.

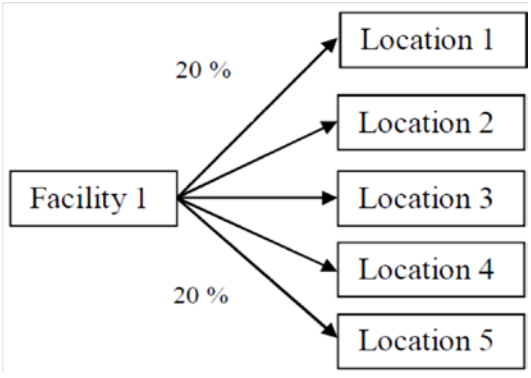

Figure 1: Random assignment for Facility 1 to locations.

In this example, over 100 replications, SimSum1 assigns facility 1 to location 5 more than other locations, because it is assumed that this assignment yields better solution according to the objective function. However we are not assured 100 per cent about such an assignment to be optimum for Problem 1, so we have other choices, i.e. facility 1 can be assigned to location 1 with the probability of 0.05, for example. Therefore, there are some chances for facility 1 to be assigned to other locations among 100 replications. As it will be shown in the

computational results section, this idea empowers the simulation model in generating potentially good random solutions for Problem 1. In the previous example, if the simulation model assigns facility 1 to any locations randomly by using a Uniform distribution then the probability of each assignment would be 20 % (Fig. 1).

Using Uniform distribution is the worst one when the goal is to find the optimum assignment and it needs a huge number of replications. Instead, SimSum1 uses better probability distribution which has some knowledge about the optimal assignment. In General, the probability of assigning facility i to location j at period t is p_{tij} in the simulation model but in SimSum1, $p_{tij} = x^*_{tij}$ where x^*_{tij} is the optimum solution of Problem 2.

SimSum1 uses simulation technique to solve a problem by generating enough numbers of random assignments. Using simulation modelling produces feasible solutions for Problem 1, easily. It is because all constraints of Problem 1 have been coded in the simulation software. Therefore, at the end of each replication, it produces a random feasible solution for the problem under consideration. On the other side, in order to improve the quality of its random solutions, it uses optimum solutions of the relaxed form of the main problem.

SimSum1 behaves similar to a Monte Carlo algorithm and does not use any time consuming processes in the simulation model. Therefore, the speed of creating a feasible solution is very fast even for large scale problems (around 0.1 sec. for a Dynamic facility layout problem with 30 facilities, 30 locations and 10 periods as the biggest problem which was solved yet). Fig. 2 depicts the general structure of SimSum1, as a flow chart.

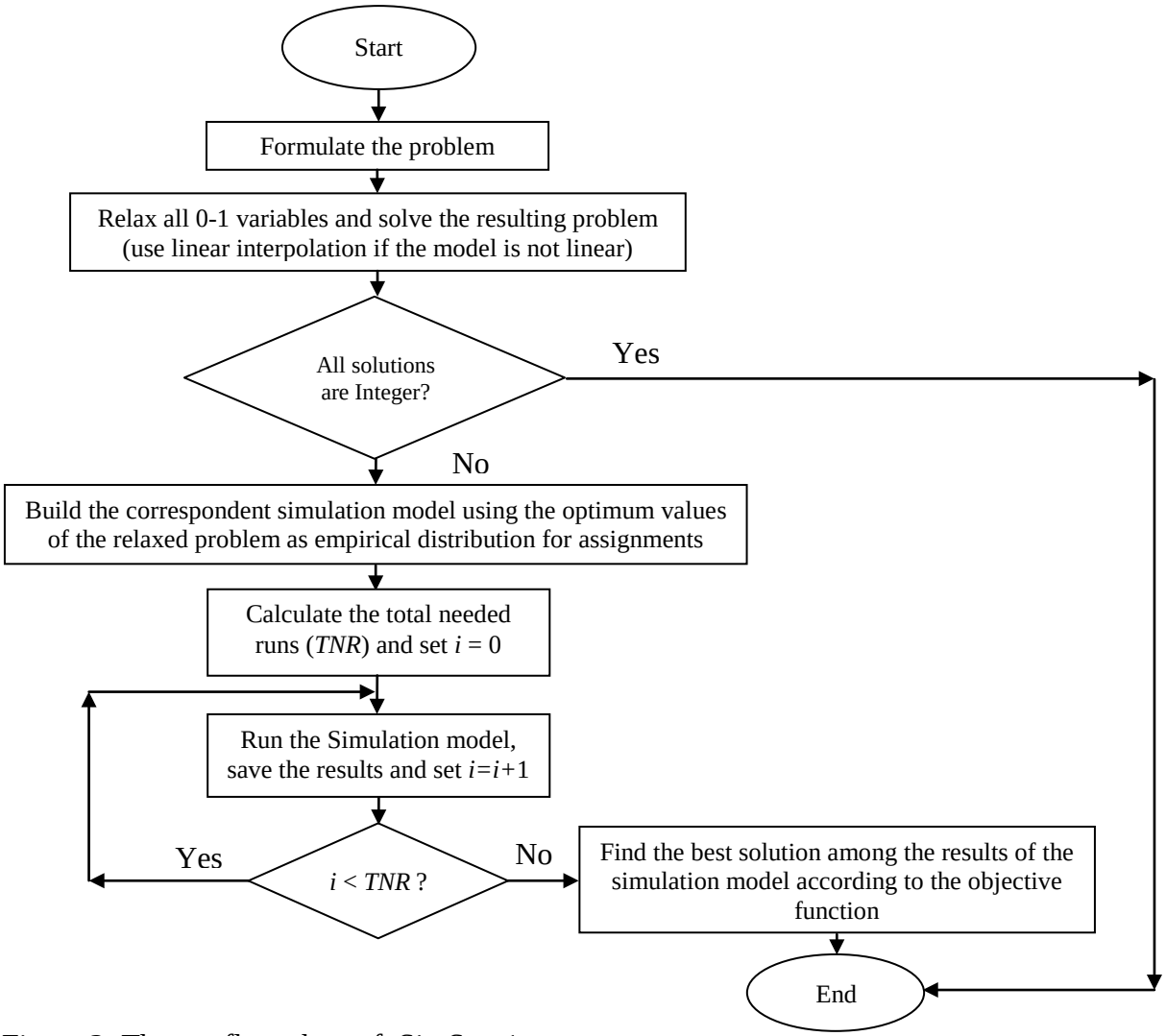

Figure 2: The top flow chart of SimSum1.

In a graph labelling problem (GLP), there is no objective function and the aim is just to obtain a feasible solution. In Problem 4, the decision variables are vertices labels and edge labels. After relaxing and solving Problem 4, there may be some non-integer values among the optimal solutions. Again, these non integer values could be interpreted as the probability of assigning a label to an edge or a vertex. It must be noticed that in contrast to the other combinatorial problems where the simulation model can produce feasible solutions, here the random generated solutions in the simulation model are not feasible, because all constraints in Problem 4 cannot be coded in the simulation model. In random assignment of vertices and edges to the labels, the simulation model can be coded in a way that each label just assigns to a vertex or an edge and vice versa. But it cannot guaranty that such assignments have the consistency property of magic sum over all vertices. Therefore among the random generated solutions in the simulation model, we need a criterion to pick up the best solution among the list. Here, it was used a specific criterion as an objective function in the simulation model. When a replication ends and the random assignment of labels to vertices and edges is realized, the simulation software calculates the following objective function (Z) for that assignment in graph $G = (V, E)$:

$$k = \left(\sum_{i\in G}\left(x_i + \sum_{(i,j)\in G} x_{ij}\right)\right)/|V| \tag{38}$$

$$Z = \sum_{i\in G}\left|\left(x_i + \sum_{(i,j)\in G} x_{ij}\right) - k\right| \tag{39}$$

where, according to definitions in section 1.3, k is the magic sum of Graph $G = (V, E)$. Since if $G = (V, E)$ has vertex-magic total labelling, for each vertex, the sum of its label plus all edge labels adjacent to it, is constant over $G = (V, E)$. Therefore, the value of k in (38) must integer and the value of Z in (39) is zero. The simulation model selects the assignment which has the minimum value of Z and if there is tie up then selects an assignment arbitrarily.

The next step in SimSum1 is to calculate the minimum number of needed runs to be assured of having the optimum solution among the random generated solutions at a certain level of significance.

3. CALCULATING THE TOTAL NEEDED RUNS

In this section, the minimum needed runs of the simulation model have been calculated. Suppose we have relaxed and solved the combinatorial problem. The optimal solutions of the relaxed problem are defined as X_{ij}^* which is the optimum probability of assigning object i to object j. For example, in DFLP it could be interpreted as probability of assigning each facility to each location at a certain period or in TSP is the probability of traveling from node i to node j or in a labelling problem it could be the probability of assigning label i to vertex/edge j. Define A_t as the probability of finding the optimum solution of the combinatorial problem in a run as follows:

$$A_t = \prod_i \prod_j X_{ij}^* \tag{40}$$

To find the optimal assignments with TNR runs in the simulation model, with % $(1-\alpha)$ as the significance level:

$$A_t + (1-A_t)A_t + (1-A_t)^2 A_t + \cdots + (1-A_t)^{TNR-1} A_t \geq 1-\alpha \tag{41}$$

Then:

$$TNR * log(1-A_t) \leq log(\alpha) \tag{42}$$

Therefore:

$$TNR \geq \frac{log\ (\alpha)}{log\ (1-A_t)} \tag{43}$$

In DFLP, since we have several periods, *TNR* must be calculated in every period and the maximum value of *TNR* among the periods will be the total needed runs. For having a conservative strategy, we must use a round up operator in calculating the *TNR*. For the relaxed DFLP, the decision variables are probabilities of assigning each facility to each location at a certain period, so *TNR* must be calculated for each period of the problem and then we must pick up the maximum *TNR* as the total needed runs. In the TSP, the decision variables are probabilities of travelling to a city from a certain city and in the Vertex-magic Total Problem; again, the decision variables are probabilities of assigning some integers (labels) to every vertex or edge.

4. COMPUTATIONAL RESULTS

Enterprise Dynamics *V8.1* software and its programming language – 4DScript- have been used for designing all simulation models, Lingo 8.0 for finding the optimal solution of the relaxed problems, and Microsoft Visual Basic 2007 as the coordinator between the simulation software and the mathematical programming software. All computations were run on a PC with a Core I6, 2.65 GHz CPU and 4GB of RAM. For DFLP with budget constraints, the same results for the largest problem which were reported by [18] have been listed in Table I.

Table I: The computational results with $\alpha = 0.01$ and $N = 30$.

N	*T*	Problem Number	Budget	Average probabilities	*NR*	Best Solution	Best Solution by Sahin et al.	% Dev.	Average Time	*N*	*T*	Problem Number	Budget	Average probabilities	*NR*	Best Solution	Best Solution by Sahin et al.	% Dev.	Average Time
15	5	P17	1	0.9306	11	481,675	481,675	0.00	0.48	30	5	P33	1	0.8870	166	576,451	577,086	-0.11	10.27
			2	0.8800	29	480,208	481,682	-0.31	1.25				2	0.9611	13	579,704	579,704	0.00	0.79
			3	0.7845	173	494,401	480,453	2.90	7.45				3	0.9256	44	577,493	577,493	0.00	2.76
		P18	1	0.8758	31	468,932	484,799	-3.27	1.35			P34	1	0.9588	14	551,951	571,846	-3.48	0.86
			2	0.7849	172	483,921	490,290	-1.30	7.39				2	0.9950	2	559,139	572,396	-2.32	0.15
			3	0.7820	182	478,213	486,726	-1.75	7.82				3	0.9009	103	556,359	570,537	-2.49	6.39
		P19	1	0.8221	85	474,661	489,583	-3.05	3.64			P35	1	0.8387	899	566,291	579,113	-2.21	55.76
			2	0.8762	31	492,274	493,018	-0.15	1.34				2	0.8735	264	556,438	579,406	-3.96	16.37
			3	0.9911	2	489,450	489,450	0.00	0.10				3	0.8870	166	566,301	574,225	-1.38	10.28
		P20	1	0.9317	11	477,414	484,876	-1.54	0.47			P36	1	0.9824	5	557,872	572,964	-2.63	0.32
			2	0.9618	6	484,856	489,912	-1.03	0.24				2	0.9731	8	554,936	578,631	-4.09	0.49
			3	0.8198	88	470,294	484,954	-3.02	3.80				3	0.9262	44	545,506	569,880	-4.28	2.70
		P21	1	0.8261	79	475,885	488,262	-2.54	3.38			P37	1	0.9224	50	552,347	559,934	-1.35	3.08
			2	0.8110	104	476,112	487,935	-2.42	4.49				2	0.9568	15	551,905	559,078	-1.28	0.92
			3	0.8148	97	469,153	487,822	-3.83	4.18				3	0.9379	29	555,069	559,506	-0.79	1.81
		P22	1	0.9153	15	473,148	486,493	-2.74	0.64			P38	1	0.9888	4	544,879	569,457	-4.32	0.23
			2	0.8581	43	473,392	488,199	-3.03	1.87				2	0.9410	26	559,640	567,166	-1.33	1.62
			3	0.9523	7	485,532	487,360	-0.37	0.30				3	0.8689	310	546,839	567,749	-3.68	19.20
		P23	1	0.9302	11	458,388	478,000	-4.10	0.48			P39	1	0.8817	199	569,470	569,470	0.00	12.33
			2	0.8334	69	466,110	487,007	-4.29	2.95				2	0.7843	6740	570,521	570,521	0.00	417.85
			3	0.8044	118	467,295	486,801	-4.01	5.08				3	0.9374	30	563,648	569,382	-1.01	1.84
		P24	1	0.8628	40	480,468	491,080	-2.16	1.71			P40	1	0.9920	3	556,582	579,411	-3.94	0.19
			2	0.9481	8	489,292	494,369	-1.03	0.33				2	0.9270	42	565,906	586,310	-3.48	2.63
			3	0.9867	3	476,618	491,237	-2.98	0.12				3	0.9413	26	560,792	577,719	-2.93	1.61

	10	P25	1	0.9476	8	939,786	981,531	-4.25	0.38		10	P41	1	0.9291	39	1,133,743	1,171,634	-3.23	2.88
			2	0.8936	23	985,031	985,031	0.00	1.10				2	0.9860	4	1,155,647	1,172,520	-1.44	0.32
			3	0.9821	3	979,638	979,638	0.00	0.16				3	0.9198	54	1,129,068	1,171,500	-3.62	3.96
		P26	1	0.9060	18	979,655	979,655	0.00	0.87			P42	1	0.9475	21	1,166,613	1,174,896	-0.71	1.52
			2	0.7792	192	955,783	981,478	-2.62	9.41				2	0.9660	11	1,137,578	1,175,998	-3.27	0.77
			3	0.9540	7	952,918	977,462	-2.51	0.33				3	0.9457	22	1,162,838	1,177,009	-1.20	1.62
		P27	1	0.9215	13	955,190	984,103	-2.94	0.65			P43	1	0.9223	50	1,169,208	1,169,208	0.00	3.63
			2	0.9272	12	972,096	993,049	-2.11	0.58				2	0.8739	260	1,179,660	1,179,660	0.00	19.01
			3	0.8726	33	960,196	983,112	-2.33	1.63				3	0.8867	167	1,134,677	1,164,129	-2.53	12.23
		P28	1	0.9512	7	950,604	971,759	-2.18	0.35			P44	1	0.9085	80	1,140,598	1,151,468	-0.94	5.81
			2	0.8484	52	974,385	974,385	0.00	2.54				2	0.7854	6,462	1,152,874	1,152,874	0.00	471.72
			3	0.9101	17	973,223	974,792	-0.16	0.81				3	0.9123	70	1,122,006	1,147,234	-2.20	5.11
		P29	1	0.7854	170	936,480	978,456	-4.29	8.34			P45	1	0.9703	9	1,114,861	1,127,044	-1.08	0.65
			2	0.9871	3	980,346	980,346	0.00	0.13				2	0.8580	453	1,141,881	1,141,881	0.00	33.09
			3	0.7638	260	947,673	978,748	-3.18	12.73				3	0.8781	225	1,128,472	1,129,703	-0.11	16.44
		P30	1	0.7820	182	949,566	970,024	-2.11	8.91			P46	1	0.8762	240	1,132,099	1,146,000	-1.21	17.55
			2	0.7930	147	972,765	972,765	0.00	7.20				2	0.7867	6,149	1,154,691	1,154,691	0.00	448.88
			3	0.7929	147	969,998	970,435	-0.04	7.22				3	0.8583	449	1,145,044	1,145,858	-0.07	32.75
		P31	1	0.7887	160	962,403	978,549	-1.65	7.83			P47	1	0.7780	8,585	1,210,573	1,210,573	0.00	626.71
			2	0.8457	55	990,976	990,976	0.00	2.67				2	0.9444	23	1,210,573	1,210,573	0.00	1.70
			3	0.7747	210	979,339	979,339	0.00	10.27				3	0.8937	132	1,210,573	1,210,573	0.00	9.62
		P32	1	0.8746	32	971,053	985,001	-1.42	1.57			P48	1	0.7786	8,389	1,199,048	1,189,154	0.83	612.37
			2	0.8432	57	958,486	986,493	-2.84	2.80				2	0.8720	278	1,152,896	1,201,885	-4.08	20.30
			3	0.9894	2	977,270	985,817	-0.87	0.12				3	0.9003	105	1,181,360	1,181,360	0.00	7.68
												Average		0.8903	465.67	790,158	802,201	-1.64	32.04
												StDev		0.0670	1,624	282,946	283,850	1.56	115.94

All parameters for the DFLP with budget constraint were taken from a data set provided by [9] which had 48 test problems (16 problems with $N = 5$ and $T = 5$ or 10, 16 problems with $N = 15$ and $T = 5$ or 10 and 16 problems with $N = 30$ and $T = 5$ or 10). For comparison, the results have been compared to those reported by [17]. In order to show the efficiency of the proposed algorithm, just the biggest problems were taken and compared (32 test problems out of 48 ones where $N = 15$, 30 and $T = 5$ or 10). The second column in this table is the problem number denoted by [17]. For each problem without a budget constraint, shows three problems with different budget constraints. First, a total budget constraint is obtained by solving an unconstrained problem with respect to the budget, and by setting the total rearrangement cost of this solution as the total budget constraint. Then the allocation of this total budget is carried out in three ways: (1) Divide the total budget by the number of periods), and allocate equally to the periods. (2) The level of the budget for each period is found by taking the half of the rearrangement cost for the same period in solution of the unconstrained problem. (3) The level of the budget for each period is found by adding 10 % more to the rearrangement costs for the same period in the solution of the unconstrained problem. Through this process three sets of problems are obtained which are denoted by 1, 2, and 3 in the third column. For the exact parameter values of the obtained problems one may refer to [9]. The forth column, labelled "Average Probabilities", lists the average optimal solutions obtained from Problem 2. The total number of needed runs (total replications executed in the simulation model) is listed in the fifth column. The optimal solutions under the proposed algorithm are listed in the sixth

column. The seventh column lists the optimal solutions reported by [17]. The percentage of deviation, denoted by "% Dev." of the best solution obtained from the proposed algorithm, which is lower than the best solution obtained from [17], is given in the eighth column, for every test problem. In the last column, the average run times are given in minutes.

As the results show, the proposed algorithm could improve the quality of near optimum solutions by 1.64 % on average, while the standard deviation of both results over 32 test problems is more or less the same. Another important criterion is that the speed of the proposed algorithm problem was about 32 minutes, on average while the CPU time was not reported in [17].

As the second evidence to approve the validity of SimSum1, several experiments were carried out on 16 big test problems derived from TSPLIB [34] which have more than 1000 cities (from 1000 to 2392 cities) and the results were compared to the one reported by [21] who used RMGA algorithm in their research. The results are shown in Table II. In this table, the first column is the name of the test problem. The second column is the number of nodes in the problem. The third column is the optimal solution. The 4^{th} and the 5^{th} columns are the best solutions found by [21] and the proposed algorithm, respectively. The 6^{th} column shows the error percentages of both algorithms and the last two columns are the CPU time in have been both algorithms in seconds. As the results show, both algorithms could find the optimum solutions but SimSum1 was about 57 % quicker than RMGA.

Table II: Computational results for the Travelling Salesman Problem with $\alpha = 0.01$.

Problem	Nodes	Optimal Solution	Best Sol. reported by [21]	Best Sol. by SimSum1	Error Percentage		Time in Sec.	
					[21]	SimSum1	[21]	SimSum1
pr1002	1002	259,045	259,045	259,045	0	0	1,442.38	485.12
u1060	1060	224,094	224,094	224,094	0	0	1,695.81	499.41
vm1084	1084	239,297	239,297	239,297	0	0	1,305.28	553.17
pcb1173	1173	56,892	56,892	56,892	0	0	1,668.02	712.64
d1291	1291	50,801	50,801	50,801	0	0	1,084.62	669.80
rl1304	1304	252,948	252,948	252,948	0	0	1,310.45	765.11
rl1323	1323	270,199	270,199	270,199	0	0	1,273.82	768.18
nrw1379	1379	56,638	56,638	56,638	0	0	2,574.60	821.82
u1432	1432	152,970	152,970	152,970	0	0	2,157.34	869.72
d1655	1655	62,128	62,128	62,128	0	0	3,435.23	951.64
vm1748	1748	336,556	336,556	336,556	0	0	1,994.38	991.46
u1817	1817	57,201	57,201	57,201	0	0	2,144.69	1,189.38
rl1889	1889	316,536	316,536	316,536	0	0	4,584.98	1,245.18
d2103	2103	80,450	80,450	80,450	0	0	2,919.70	1,482.76
u2152	2152	64,253	64,253	64,253	0	0	3,198.70	1,762.43
pr2392	2392	378,032	378,032	378,032	0	0	4,243.70	1,921.53
Average		178,628	178,628	178,628	0	0	2,314.61	980.58

As the third evidence, the results of applying SimSum1 over a range of vertex-magic labelling problems have been listed in Table III. For comparison, the results were compared to the ones reported by [29] over some specified test problems. However, because a GLP has not any objective function, this comparison just includes the speed of both algorithms not solution qualities. Naughty 2.0 software has been used to generate a random graph with desired number of edges and vertices. For each class of graphs, 30 random samples were generated by

the graph generator software and the average computational result was recorded for that class. In this table, the first column is the type of the graph. The second column is the number of edges and the third column is the number of vertices. The 4th and the 5th columns are the number of constraints and variables in the mathematical model. The 6th column indicates that the graph has the vertex-magic total labelling or not. The 7th and the 8th columns are the CPU time reported by [29] and the proposed algorithm, respectively. So, in Table III, graphs which are VMTL are included (or $Z = 0$ according to (39)). According to the results in Table III, the proposed algorithm is about 52 % faster than the one developed by [29]; on average and this is the only available criterion to compare these algorithms in the absence of any objective functions.

Table III: The results of the algorithm for different class of graphs with $\alpha = 0.01$.

Graph type		*m*	*n*	Number of variables	Number of Constraints	VMTL?	Average time of [29] in seconds	Average time of proposed algorithm in seconds
Cycles	C15	15	15	931	105	Yes	23.26	0.15
	C20	20	20	1,641	140	Yes	68.46	0.25
	C25	25	25	2,551	175	Yes	452.45	132.48
	C30	30	30	3,661	210	Yes	886.02	411.69
	C35	35	35	4,971	245	Yes	1,853.79	955.2
	C40	40	40	6,481	280	Yes	4,205.33	2,698.11
	C45	45	45	8,191	315	Yes	9,541.12	5,924.18
Snakes	P15	14	15	871	102	Yes	13.1	0.11
	P20	19	20	1,561	137	Yes	54.23	12.65
	P25	24	25	2,451	172	Yes	385.33	148.74
	P30	29	30	3,541	207	Yes	745	365.82
	P35	34	35	4,831	242	Yes	1,643.89	965
	P40	39	40	6,321	277	Yes	3,965.21	2,411.08
	P45	44	45	8,011	312	Yes	9,002.27	5,620.44
Complete Graphs	K10	45	10	3,081	175	Yes	645.21	485.46
	K20	190	20	44,311	650	Yes	28,747.45	17,001.33
Complete Bipartite Graphs	K5,5	25	10	1,261	115	Yes	30.98	1.05
	K10,10	100	20	1,4521	380	Yes	15,887.74	8,111.57
Wheels	W10	20	11	993	104	Yes	29.31	0.21
Generalized Peterson Graphs	P(5,2)	15	10	651	85	Yes	14.25	0.16
	P(8,4)	20	16	1,333	124	Yes	76.2	1.47
	P(10,5)	25	20	2,071	155	Yes	1,034.53	753.17
Product graphs	K4×P10	46	20	4,423	218	Yes	17,855.32	16,385.92
Average							4,224.37	2,712.45

To extend the range of graph classes, the results of using SimSum1 over a wide range of large scale trees have been listed in Table IV. The results show that the algorithm can find the

magic labels in reasonable times even for a tree with 70 vertices (as the biggest tree for which the magic labels has been reported, yet).

Table IV: The results of the algorithm for tree graphs with significant level 99 %.

m	*n*	Number of variables	Number of constraints	VMTL?	Average time of proposed algorithm in seconds
19	20	1,561	137	Yes	118.26
29	30	3,541	207	Yes	318.60
39	40	6,321	277	Yes	1,856.08
49	50	9,901	347	Yes	2,860.21
59	60	14,281	417	Yes	4,378.69
69	70	19,461	487	Yes	9,654.72
Average					3,197.76

All three examples have been selected from the well-known combinatorial optimisation problems which have wide range of applications. As the results show, SimSum1 has enough accuracy and speed to solve these problems even for the large-scale ones. However, because the power of the basic computer affects the computational time, it is hard to realize the part of speed improvements that made by SimSum1. However, the used computer is a very normal computer in nowadays market and at least, it can be inferred that the proposed algorithm had reasonable speed.

5. CONCLUSIONS

In this research, a new general heuristic algorithm named SimSum1 has been introduced for solving a class of combinatorial problems which the decision variables are 0-1 and the summation of all decision variables is equal to 1. SimSum1 combines discrete even simulation together with relaxation techniques to solve this kind of combinatorial problems. The proposed algorithm not only avoids common issues in meta-heuristic algorithms such as premature events, parameter tuning and trapping in local optimums but also uses a simulation technique that produces feasible solutions without the use of any specific non-realistic assumptions. On the other side, the simulation model does not depend on the simulation clock so generating a feasible solution is a very fast process which is another advantage of using the simulation technique. The class of combinatorial problems with the mentioned properties have 0-1 variables which must be relaxed to have a linear problem. SimSum1 inputs the optimum solutions of the relaxed problem to the simulation model as an empirical distribution for assignments inside the simulation model. This idea as the main contribution of the paper can improve the performance of the simulation to obtain better results. For showing the efficiency of SimSum1, the algorithm was tested over several well-known combinatorial problems like Dynamic Facility Layout Problem, Travelling Salesman Problem and Graph Labelling Problem. As the results of experiments show SimuSum1 can solve complex combinatorial problems efficiently, even large-scale sizes. The results show that at least for DFLP and GLP, SimSum1 is the fastest heuristic algorithm developed so far in the literature which is the second contribution of the current research. Regarding the constraints, inherent in this kind of research, we think that if we use the new version of Lingo software and run the algorithm on a faster computer (in particular, one with a faster CPU) the results would be

further improved. Although the SimSum1 has a very good performance, but its main weakness point is that its application is restricted to integer mathematical problems which have the special constraint. Finally, for the future researches, the suggestions are to combine other evolutionary algorithms such as Particle Swarm Optimisation or Ant Colony Optimisation with the simulation technique to obtain possible better results and also extending the application of SimSum1 in other famous problems such as Assembly Line Balancing Problems or Generalized Assignment Problem.

REFERENCES

[1] Drira, A; Pierreval, H.; Hajri-Gabouj, H. (2007). Facility layout problems: A survey, *Annual Reviews in Control*, Vol. 31, No. 2, 255-267, doi:10.1016/j.arcontrol.2007.04.001

[2] Tompkins, J.; White, J.; Bozer, Y.; Tanchoco, J. (2003). *Facilities planning*, 3rd ed., John Wiley & Sons, New Jersey

[3] Dong, M., Wu; C., Hou; F. (2009). Shortest path based simulated annealing algorithm for dynamic facility layout problem under dynamic business environment, *Expert Systems with Applications*, Vol. 36, No. 8, 11221-11232, doi:10.1016/j.eswa.2009.02.091

[4] Gary, Y.; Chen, K. J.; Rogers, A. (2009). Multi-objective evaluation of dynamic facility layout using ant colony optimization, *Proceedings of the 2009 Industrial Engineering Research Conference*

[5] Page, A. L. (1991). New product development survey: Performance and best practices, *Proceedings of PDMA conference*

[6] Balakrishnan, J.; Cheng, C. H. (2009). The dynamic plant layout problem: Incorporating rolling horizons and forecast uncertainty, *Omega*, Vol. 37, No. 1, 165-177, doi:10.1016/j.omega.2006.11.005

[7] Kouvelis, P.; Kurawarwala, A. A.; Gutierrez, G. J. (1992). Algorithms for robust single and multiple periods layout planning for manufacturing systems, *European Journal of Operations Research*, Vol. 63, No. 2, 287-303, doi:10.1016/0377-2217(92)90032-5

[8] Rosenblatt, M. J. (1986). The dynamics of plant layout, *Management Science*, Vol. 32, No. 1, 76-86, doi:10.1287/mnsc.32.1.76

[9] Balakrishnan, J.; Cheng, C. H. (2000). Genetic search and the dynamic layout problem: An improved algorithm, *Computers and Operations Research*, Vol. 27, No. 6, 587-593, doi:10.1016/S0305-0548(99)00052-0

[10] Kaku, B. K.; Mazzola, J. B. (1997). A tabu-search heuristic for the dynamic plant layout problem, *INFORMS Journal on Computing*, Vol. 9, No. 4, 374-384, doi:10.1287/ijoc.9.4.374

[11] Baykasoglu, A.; Gindy, N. N. Z. (2001). A simulated annealing algorithm for dynamic layout problem, *Computers and Operations Research*, Vol. 28, No. 14, 1403-1426, doi:10.1016/S0305-0548(00)00049-6

[12] Balakrishnan, J.; Cheng, C. H.; Conway, D. G.; Lau, C. M. (2003). A hybrid genetic algorithm for the dynamic plant layout problem, *International Journal of Production Economics*, Vol. 86, No. 2, 107-120, doi:10.1016/S0925-5273(03)00027-6

[13] Erel, E.; Ghosh, J. B.; Simon, J. T. (2003). New heuristic for the dynamic layout problem, *Journal of the Operational Research Society*, Vol. 54, 1275-1282, doi:10.1057/palgrave.jors.2601646

[14] McKendall, A. R; Shang, J. (2006). Hybrid ant systems for the dynamic facility layout problem, *Computers & Operations Research*, Vol. 33, No. 3, 790-803, doi:10.1016/j.cor.2004.08.008

[15] McKendall, A. R.; Shang, J.; Kuppusamy, S. (2006). Simulated annealing heuristics for the dynamic facility layout problem, *Computers & Operations Research*, Vol. 33, No. 8, 2431-2444, doi:10.1016/j.cor.2005.02.021

[16] Balakrishnan, J.; Cheng, C. H. (1998). Dynamic layout algorithms: A state-of-the art survey, *Omega*, Vol. 26, No. 4, 507-521, doi:10.1016/S0305-0483(97)00078-9

[17] Sahin, R.; Ertogral, K.; Turkbey, O. (2010). A simulated annealing heuristic for the dynamic facility layout problem with budget constraint, *Computers & Industrial Engineering*, Vol. 59, No. 2, 308-313, doi:10.1016/j.cie.2010.04.013

[18] Azimi, P.; Charmchi, H. M. (2011). A new heuristic algorithm for the dynamic facility layout problem with budget constraint, *Proceedings of International Conference on Modelling and Simulation (ICMS 2011)*, Amsterdam, The Netherlands
[19] Applegate, D. L.; Bixby, R. E.; Chvatal, V.; Cook, W. J. (2006). *The traveling salesman problem: A computational study*, Princeton University Press, Princeton, USA
[20] Garey, M. R.; Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco, USA
[21] Liu, F.; Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the TSP, *Expert Systems with Applications*, Vol. 36, No. 3, 6995-7001, doi:10.1016/j.eswa.2008.08.026
[22] Azimi, P.; Daneshvar Ghorbani, P. (2010). An Efficient Heuristic Algorithm for the Traveling Salesman Problem, *Proceedings of the 8th International Heinz Nixdorf Symposium*, Paderborn, Germany
[23] Rosa, A. (1967). On certain valuations of the vertices of a graph, *Theory of Graphs – Int. Symposium*, Rome, Gordon and Breach, New York and Dunod Paris, 349-355
[24] Graham, R. L.; Sloane, N. J. A. (1980). On additive bases and harmonious graphs, *SIAM Journal on Algebraic Discrete Methods*, Vol. 1, No. 4, 382-404
[25] Gallian, J. A. (2009). A dynamic survey on graph labelling, *The Electronic Journal of Combinatorics*, Vol. 16, #DS6
[26] Golomb, S. W.; Bloom, G. S. (1977). Application of numbered undirected graphs, *Proceedings of the IEEE*, Vol. 65, No. 4, 562-570
[27] Redl, T. A. (2003). Graceful graphs and graceful labellings: two mathematical programming formulations and some other new results, *Technical Report TR03-01*, CAAM Department, Rice University, Texas, USA
[28] Eshghi, K.; Azimi, P. (2004). Applications of mathematical programming in graceful labelling of graphs, *Journal of Applied Mathematics*, Vol. 1, 1-8, doi:10.1155/S1110757X04310065
[29] Eshghi, K.; Azimi, P. (2007). An algorithm for finding a feasible solution of graph labelling problems, *Utilitas Mathematica*, Vol. 72, 163-174
[30] Azimi, P. (2010). An application of mathematical programming in vertex-magic total labelling problems, *Far East Journal of Mathematical Sciences*, Vol. 42, No. 2, 161-173
[31] Golomb, S. W. (1972). How to number a graph, Read, R. C. (Ed.), *Graph Theory and Computing*, Academic Press, New York, 23-37
[32] MacDougall, J. A; Miller, M.; Wallis, W. D. (2004). Vertex magic total labelling of graphs, *Proceedings Australasian Workshop Combinatorial Algorithm*, 222-229
[33] Bosák, J. (1990). *Decompositions of Graphs*, Kluwer Academic Publishers Group, Dordrecht
[34] Reinelt, G. (1991). TSPLIB: A travelling salesman problem library, *INFORMS (ORSA) Journal on Computing*, Vol. 3, No. 4, 376-384, doi:10.1287/ijoc.3.4.376