2010

An Empirical Comparison of Improvement Heuristics for the Mixed-Model, U-Line Balancing Problem

John K. Visich
Bryant University

Basheer M. Khumawala
University of Houston

Joaquin Diaz-Saiz
University of Houston

Recommended Citation
International Journal of Manufacturing Technology and Management, Vol. 20, Nos. 1/2/3/4, 2010, pp. 25-45.

An Empirical Comparison of Improvement Heuristics for the Mixed-Model, U-Line Balancing Problem

*John K. Visich, Bryant University, 1150 Douglas Pike, Smithfield, RI 02917, jvisich@bryant.edu, 401-232-6437, 401-232-6319 (fax)

Basheer M. Khumawala, C.T. Bauer College of Business, University of Houston, Houston, TX, 77204, bkhumawala@uh.edu, 713-743-4721, 713-743-4940 (fax)

Joaquin Diaz-Saiz, C.T. Bauer College of Business, University of Houston, Houston, TX, 77204, jdiaz-saiz@uh.edu, 713-743-4713, 713-743-4940 (fax)

*corresponding author

John Visich is an associate professor in the Management Department at Bryant University where he teaches courses in operations management, supply chain management, and international operations. He has a Ph.D. in Operations Management from the University of Houston, where he received the Melcher Award for Excellence in Teaching by a Doctoral Candidate. His research interests are in supply chain and health care applications of radio frequency identification, supply networks, and U-shaped assembly lines. He has published in *Journal of Managerial Issues*, *International Journal of Integrated Supply Management*, *Sensor Review*, *International Journal of Healthcare Technology and Management* and others.

Basheer Khumawala is John & Rebecca Moores Professor and Chair of the Decision and Information Sciences Department at the University of Houston where he teaches courses in Supply Chain Management. His Ph.D. is from Purdue, and his teaching areas are production operations and logistics management. He has previously taught at UNC-Chapel Hill, Purdue, Rice and other Universities overseas. His publications have appeared in *Management Science*, *Naval Research Logistics Quarterly*, *AIIE Transactions*, *Journal of Operations Management*, *Production and Inventory Management*, *Sloan Management Review* and others. He is a Fellow of the Decision Sciences Institute and the Pan Pacific Business Association.

Dr. Diaz-Saiz joined the faculty at the University of Houston in the fall of 1985. He received his doctorate in Statistics from Oklahoma State University and has articles published in journals such as *Annals of Statistics*, *Communications in Statistics*, *Journal of Statistical Planning and Inference*, *International Journal of Forecasting*, *and Estadística*. He is currently associate editor of Communications in Statistics. Dr. Díaz-Sáiz has participated in projects for a wide variety of firms in the public and private sectors. His research interests include Bayesian forecasting, inventory control, and time series analysis.

An Empirical Comparison of Improvement Heuristics for the Mixed-Model, U-Line Balancing Problem

Abstract

Mixed-model assembly lines often create model imbalance due to differences in task times for the different product models. Smoothing algorithms guided by meta-heuristics that can escape local optimums can be used to reduce model imbalance. In this research we utilize the meta-heuristics tabu search (TS), the great deluge algorithm (GDA) and record-to-record travel (RTR) to reduce three objective functions: the absolute deviation from cycle time, the maximum deviation from cycle time, and the sum of the cycle time violations. We found that the GDA was significantly superior to the RTR and TS algorithms across all problem sizes and objective functions. For the 19 task problems, RTR performed significantly better than TS for all three objective functions. On the other hand, for the 61 and 111 task problems TS performed significantly better than RTR for all three objective functions.

Key Words: Mixed-Model, U-Line, Great Deluge Algorithm, Record-to-Record Travel, Tabu Search

1 Introduction

The explosive growth of today's information based society has led to an increased consumer awareness of the purchasing options available to them and has caused an increase in consumer demand for product variety. This has put pressure on manufacturing firms to provide constant innovation as a way to remain competitive and has led to shortened product life cycles (Simatupang and Sridharan, 2002) and increased supply chain complexity in the trade-off conflict between inventory, transportation and warehousing costs versus customer service levels (Simchi-Levi, Kaminsky, and Simchi-Levi, 2000). In an effort to meet the increase in demand for product variety in order to maintain or increase revenue and mitigate the negative effects of product variety, many manufacturers have altered their production processes to include the

tactical production strategies of mass customization and just-in-time (JIT). On a company-wide strategic level, the integration of the firms supply chain improves the coordination of JIT and mass customization manufacturing systems, and allows for quicker response to changes in demand.

Adapting quickly to the market requires flexibility in both equipment and employees, and for manufacturers that utilize an assembly operation, a U-shaped line can offer advantages over a serial line layout (a straight line layout). These include improved communication between workers and the ability to adjust the production rate by removing or adding workers (Monden, 1998; Wantuck, 1989). To meet the demand for product variety many manufacturers are converting their production lines from a single product or batch production to mixed-model production. Benefits of mixed-model production are the ability to provide customers with a variety of products in a timely and cost effective manner (Sparling and Miltenburg, 1998). This research utilizes a U-shaped assembly line layout for mixed-model production.

The optimal solution to the mixed-model, U-shaped assembly line balancing problem is dependent on both the assignment of tasks to workstations and the model sequence. The mixed-model assembly line problem requires solutions to the following two problems (Ghosh and Gagnon, 1989):

1. The mixed-model line balancing problem: How will tasks be assigned to workstations?
2. The mixed-model sequencing problem: In what sequence will units of different models be produced on the line?

This research focuses on the first problem, the assignment of tasks to workstations for a given sequence of models. Three meta-heuristics methods are used to guide an algorithm that smoothes the initial balance of a mixed-model, U-shaped assembly line: tabu search (TS), the

great deluge algorithm (GDA) and record-to-record travel (RTR). We test a variety of problem sizes and subtypes, and for each line that we smooth we minimize three objective functions.

Our paper is organized as follows. In the following section 2 we review the relevant literature on U-shaped assembly line balancing. We discuss our research methodology, objective functions and problem instances in section 3. Next, in section 4, we describe the three heuristics utilized in this research and the selection of the algorithm parameters used in the empirical experiments. In section 5 we state our research questions and present our empirical results. In section 6 we conclude with a summary of our findings, discuss the limitations of our study and provide suggestions for future research.

2 U-Shaped Assembly Line Balancing Literature Review

A small, but rapidly growing, body of literature exists for U-shaped production lines, and the research can be classified into two groups: production flow lines and line balancing (Erel, Sabuncuoglu, and Aksu, 2001). In line flow research the emphasis is on identifying critical design factors and their impact on the performance of the U-line. In line balancing the objective is to minimize the cycle time, the number of workstations or in the case of the mixed-model U-line, to smooth model imbalance. Since the focus of this study is the U-shaped assembly line balancing problem (UALBP) with deterministic task times our literature review covers deterministic line balancing research. For discussions on various aspects of line flow research see Aase, Olson, and Schniederjans (2004), Celano et al. (2004), Chand and Zeng (2001), Cheng, Miltenburg, and Motwani (2000), Miltenburg (2000; 2001a; 2001b), Nakade and Ohno (1995; 1997; 1999; 2003), Nakade, Ohno, and Shanthikumar (1997), and Ohno and Nakade (1997).

Miltenburg (1998) attributed the first discussion in the open literature in English concerning U-lines to Schonberger (1982) who noticed a preference among Japanese manufacturers for multiple U-lines, where workstations often spanned more than one U-line. Additional early discussions of U-lines were by Hall (1983), Monden (1993) and Wantuck (1989).

Miltenburg and Wijngaard (1994) were the first to compare a U-shaped assembly line with a serial assembly line. They used two methods developed for the traditional single-model, serial line ALBP to solve a Type-1 UALBP (given the cycle time c, minimize the number of workstations K). An integer programming formulation to solve the Type-1 problem for the UALBP was presented by Urban (1998). This formulation used a "phantom" network to move forward and backward through the network. Other line balancing procedures for the UALBP include ULINO by Scholl and Klein (1999), U-OPT by Aase (2003), a shortest route formulation by Gökcen et al. (2005) and a goal programming approach by Gökcen and Ağpak (2006). A genetic algorithm procedure to balance U-lines is presented by Ajenblit and Wainwright (1998), while simulated annealing is used by Erel, Sabuncuoglu and Aksu (2001) and Baykasoğlu (2006).

Miltenburg (1998) analyzed the U-line facility problem where a multi-line station may include tasks from two adjacent U-lines. This extension of the basic single U-line is known as an *N* U-line facility, where *N* is the number of U-lines that are to be simultaneously balanced. Sparling (1998) and Chiang, Kouvelis, and Urban (2007) also investigated the multiple U-line problem.

The first mixed-model U-line balancing problem (M-UALBP) was addressed by Sparling and Miltenburg (1998). They adapted the four-step mixed-model, serial-line procedure of Thomopolous (1967, 1970) and set the initial balance using a branch and bound algorithm

developed for serial lines. A smoothing algorithm using a search procedure is then used to reduce the imbalance of the line for a given sequence of models. Kim, Kim, and Kim, (2000) and Kim, Kim, and Kim (2006) applied genetic algorithms to the mixed-model, U-shaped line balancing and sequencing problem.

3 Research Methodology

One of the primary differences between serial lines and U-shaped lines in a mixed-model assembly environment occurs when a U-line has a cross-over station, and hence an operator can work on two different product models during the same production cycle. This unique characteristic of a U-line layout increases the complexity of the mixed-model algorithm since the total task time in a workstation during a cycle may include work performed at both the front of the U-line and the back of the U-line. We present our algorithm notation and then our three mixed-model objective functions to be minimized. We base our notation on the work of Scholl (1999) and Sparling and Miltenburg (1998), and we make modifications specific to our representation of the problem. We define the following notation.

Inputs that are Fixed

c cycle time or launch interval (seconds)

I number of tasks, index i = 1, …, I

K number of workstations, index: k = 1, …, K

M number of product models, index: m = 1, …, M

N_m number of units of product model m in the sequence

S number of cycles, index: s = 1, …, S

$$S = \sum_{m=1}^{M} N_m$$

t_{im} processing time of task i on product model m

mf_k^s product model produced on the front of workstation k at the s-th cycle

mb_k^s product model produced on the back of workstation k at the s-th cycle

Inputs that are Variable

IF_k set of tasks at workstation k located on the front of the U-line

IB_k set of tasks at workstation k located on the back of the U-line

Calculation

T_{ks} amount of work assigned to workstation k at the s-th cycle

$$T_{ks} = \sum_{i \in IF_k} t_{imf_k^s} + \sum_{i \in IB_k} t_{imb_k^s}$$

The inputs IF_k and IB_k are variable because the smoothing algorithm swaps tasks between workstations in an attempt to reduce model imbalance. Only feasible swaps are accepted, and if so then T_{ks} is calculated for each workstation for each model cycle.

In our research we minimize three mixed-model deterministic assembly line balancing objective functions. The first objective function is the sum of the absolute deviation from cycle time (ADC) and it was first introduced by Thomopolous (1970) for a serial line layout. Recently it has been tested empirically by Bukchin (1998) for a serial line layout, and for a U-line layout by Sparling and Miltenburg (1998) and Kim, Kim, and Kim (2000; 2006). Our second objective function is the maximum deviation from cycle time (MDC) (Scholl, 1999). Our third objective function is the sum of the cycle time violations (SCV) (Scholl, 1999; Sparling and Miltenburg, 1998). To our knowledge, neither the MDC nor the SCV have been tested empirically in a U-line layout. For our three mixed-model objective functions we again base our notation on the work of Scholl (1999) and Sparling and Miltenburg (1998), and we make modifications specific to our representation of the problem. We define the following objective functions:

ADC: sum of the absolute deviation from cycle time

Objective 1: Minimize $ADC = \sum_{k=1}^{K} \sum_{s=1}^{S} |T_{ks} - c|$

MDC: maximum deviation from cycle time

Objective 2: Minimize $MDC = \max\{|T_{ks} - c|\}$

SCV: sum of the cycle time violations.

Objective 3: Minimize $SCV = \sum_{k=1}^{K} \sum_{s=1}^{S} \max(0, T_{ks} - c)$

For each simulation we run to minimize an objective function we record the initial and final objective function values. In the next section we discuss the minimum part set which directly impacts the number of cycles (S) that the objective functions evaluate.

3.1 Minimum Part Set and Unique Sequences

Solution approaches to the mixed-model assembly line balancing problem use either the full part set (Thomopolous, 1970; Dar-El and Cother, 1975) or the minimum part set (Bard, Dar-El, and Shtub, 1992; Bard, Shtub, and Joshi, 1994; Kim, Kim, and Kim, 2000; 2006). The full part set uses the total demand for each product model over the planning horizon (usually a single work shift). Tasks times are based on a weighted average of the times to perform a specific task for each product model, which often results in fractional tasks times for computations. The minimum part set (MPS) is the smallest part set having the same product model proportion as the total demand. For example, if we produce three product models (Model A, Model B and Model C) and our total demand over the planning horizon is 60 units of Model A, 40 units of Model B and 20 units of Model C, we determine the highest common divisor for all three product model demands. In this example that divisor is 20 and we divide the demand of each product model by 20. This gives 3 units of Model A, 2 units of Model B and 1 unit of Model C or an MPS of 321. Bard et al. (1992) point out that production schedules based on the MPS are more manageable

than a schedule based on the full part set, and that the MPS approach greatly simplifies computations. In addition, McCormick et al. (1989) have shown that MPS based schedules quickly reach a steady state.

Thomopoulos (1967) shows that from combinatorial analysis the total number of possible product model sequences is:

$$\frac{N!}{N_A!N_B!N_C!\ldots}$$ where $N = N_A + N_B + N_C + \ldots$, and $N_A, N_B, N_C, \ldots$ are the number of units of product models A, B, C, … to be produced. In the above formula, the number of sequences increases as the number of product models and units of each product model increases. In the above example demonstrating the derivation of the MPS, our MPS of 321 has a total of 60 possible sequences [6! ÷ (3!*2!*1!)]. But, when using the MPS, only the unique sequences need to be evaluated. The number of unique sequences for a given MPS is the total number of sequences divided by the total number of units in the MPS. For our example, the number of unique sequences is 60 ÷ (3 + 2 + 1) = 10 unique sequences. For an MPS = 111 (based on one unit each of product models A, B and C) there will be 3! ÷ (1!*1!*1!) = 6 sequences of which 6 ÷ (1+1+1) = 2 will be unique: ABC and ACB. Sequences BCA and CAB are not unique since they are equivalent to ABC, and sequences CBA and BAC are not unique since they are equivalent to ACB. In this research we test two unique sequences for a given MPS. These sequences were selected by using Excel to assign a random number to each unique sequence and then selecting the two sequences with the lowest random numbers.

3.2 Balancing Procedure Steps and Illustrated Example

Our balancing procedure for the mixed-model assembly line balancing problem is based on the four-step heuristic procedure proposed by Thomopolous (1967; 1970) for a serial line. This procedure was used by Sparling and Miltenburg (1998) for the M-UALBP and hence provides

our motivation for using this procedure in our research. Since the Thomopolous (1967; 1970) procedure uses the full part set, we will use a modified version to accommodate our use of the minimum part set. A smoothing algorithm for the M-UALBP using the minimum part set is as follows:

Step 1. For each task, multiply the task time for the product model by the number of units of the product model in the sequence and sum the total task times for all the product models. This is our total task time for a task.

Step 2. Merge each product models precedence diagram into a single precedence graph.

Step 3. Multiply the desired cycle time by the total number of units of product models in the sequence (S from our notation above) and use this value as the cycle time. Solve a Type-1, single-model assembly line balancing problem with the tasks and total task times from Step 1 and the merged precedence diagram from Step 2. In this research we use ULINO (Scholl and Klein, 1999). The solution is our initial balance.

Step 4. Smooth the initial balance from Step 3 to reduce model imbalance using one of the three objective functions previously presented. Use the heuristic search techniques discussed in the next section to prevent the smoothing algorithm from becoming trapped in a local optimum by allowing exchanges that increase model imbalance.

3.3 Problem Instances, Data Sets and Research Assumptions

Scholl (1999) distinguishes between the *problem* (also called problem type) and *problem instance* for the assembly line balancing problem (ALBP). *Problem* refers to the type of assembly line balancing problem to be solved and is based on the four primary ALBP classifications and the three objective function subtypes (Ghosh and Gagnon, 1989). Problem classifications are single model or multi/mixed model with either deterministic or stochastic task times. Objective function subtypes are:

- Type - 1: given the cycle time c, minimize the number of workstations K.
- Type - 2: given the number of workstations K, minimize the cycle time c.
- Type - 3: minimize or maximize an objective function by varying c and K.

In our research the *problem* we are solving is the mixed-model deterministic ALBP in a U-shape. We initially solve a Type-1 objective function subtype and then through the smoothing algorithm we solve a Type-3 objective function subtype. Minimizing one of our three objective functions also tends to minimizes the effective cycle time.

Problem instances are those specific values for all *problem* parameters and can be fixed or variable. Fixed *problem instances* are those characteristics specific to a mixed-model data such as the number of tasks, the number of product models, the tasks times for each task for each product model, and the task precedence relationships for each of the product models. Variable characteristics of a *problem instance* include cycle time, number of workstations, minimum part set (MPS), and the unique sequences associated with a specific MPS. In this research we test a variety of these variables in order to cover a wide range of *problem instances*.

Three different data sets from the literature are used in this research and are shown in Table 1. The 19-Task, 3-Model data set can be found in Thomopolous (1970), and was used by Sparling and Miltenburg (1998) in a mixed-model, U-line layout example to demonstrate a smoothing algorithm. In our research we multiplied all Thomopolous task times by 10 to eliminate fractional task times, which eased program verification. The 61-Task, 4-Model data set comes from Kim, Kim, and Kim (2000) and the 111-Task, 5-Model data set comes from Arcus (1963). Kim, Kim, and Kim (2000) tested all three data sets in their empirical study.

Table 1 Experiment Data Sets

Data Name	Number of Tasks	Number of Models	Maximum Task Time (seconds)	Code
Thomopolous	19	3	15	T
Kim	61	4	34	K
Arcus	111	5	*6,615	A

* Following Kim, Kim, and Kim (2000) the processing time for task 95 is changed from 33491 to 6615 seconds to allow for a larger number of workstations for a given cycle time.

In our solution algorithm for the mixed-model, U-shaped assembly line balancing problem we make several assumptions. Our assumptions come primarily from Sparling and Miltenburg (1998) since their research also focused on the M-UALBP and also from Thomopolous (1967; 1970) and Scholl (1999). The assumptions made in this research are:

- precedence diagrams can be combined
- task times are deterministic
- task times may be different for different product models
- each task type is assigned to only one station regardless of models
- processing time equals task time
- tasks may not be split
- cycle time equals launch rate
- the line is paced
- workstations are closed
- the workforce is multi-skilled and flexible
- travel time equals zero
- task locations are not fixed

4 Heuristic Development

The heuristics we propose to test to reduce model imbalance are tabu search (Glover, 1977), the great deluge algorithm and record-to-record travel (Dueck, 1993). All three heuristics will be implemented in an improvement formulation and we discuss them in the following sections.

4.1 Tabu Search

Tabu search is now a well known meta-search heuristic introduced by Glover (1977) that employs a search strategy to accept inferior solutions in order to escape local optimums. Tabu search starts with a random, feasible solution to the problem and from this solution a set of neighboring solutions is generated. A neighbor solution is generated through a pre-defined change (known as a move) to the incumbent solution such that the resulting solution is feasible. The quality of each solution is evaluated using a specified cost function and the best solution in the current set of neighboring solutions is selected as the new incumbent solution. A new set of neighboring solutions is then generated and the process repeats until a stopping condition is met.

Without modification, this process can become trapped in a local optimum. Therefore, tabu search utilizes a flexible short-term memory of recent moves known as the tabu list. With a tabu list, the selection of the new incumbent solution is the best neighboring solution according to the cost function whose generating move is not on the tabu list. This strategy prevents backtracking into local optima and can force the acceptance of inferior solutions that might lead to better solutions. The length of the tabu list is critical since it determines the length of time moves remain unavailable. A list that is too long will restrict the moves available and a list that is too short will result in a cycling of solutions. If a move on the tabu list results in a solution better than the best one so far, the move's tabu status is ignored and the solution is immediately accepted. This is known as aspiration criteria.

4.2 The Great Deluge Algorithm

The great deluge algorithm (Dueck, 1993) is based on the general purpose optimizing algorithm threshold accepting, which was first developed by Dueck and Scheuer (1990). Threshold accepting in turn is based on simulated annealing, and though both heuristics have similar convergence properties, they have different acceptance rules. The great deluge algorithm (GDA) is analogous to a person who needs to find the highest point of land during a deluge. As the water level rises, the algorithm moves around the land (feasible region) until it reaches a high point. The water rises according to a rain speed (labeled UP) which is similar to the temperature parameter in simulated annealing. For the ALB problem, we want to minimize the imbalance between stations. Therefore UP will be more like a leak rate and we will lower the water level.

The GDA starts with an initial feasible solution, and starting values for the rain speed parameter and water level parameter (initial objective function value), both of which must be greater than zero. A new solution is chosen based on a stochastic perturbation of the old solution

and the function value of the new solution is calculated. If the function value of the new solution is greater than the function value of the old solution, the old solution becomes the new solution, the water level is decreased and the process repeats until there is no longer a cost decrease or until a specified termination point is reached. If the new solution is less than the old solution, the new solution is kept, the water level is decreased and the process repeats. The rain speed parameter is critical because it impacts both the computation speed and the quality of the results. If UP is too high the algorithm works very quickly, but solution quality will be poor. If UP is very low, then the solution quality will be much better, but the computation time will take longer (Dueck, 1993).

4.3 Record-to-Record Travel

Record-to-record travel (Dueck, 1993) is also based on threshold accepting (Dueck and Scheuer, 1990) and it is very similar to the GDA. The rate at which the water level changes is linked to the rate at which the solution improves. The water level in the GDA becomes the value of the record (R) in record-to-record travel (RTR) and the rain parameter UP becomes the deviation parameter (D). The selection of the deviation parameter affects the results the same way as the rain parameter. The difference between the two heuristics is in the acceptance criteria.

Record-to-record travel starts with an initial feasible solution, and starting values for the deviation parameter and the record (initial objective function value), both of which must be greater than zero. A new solution is chosen based on a stochastic perturbation of the old solution and the function value of the new solution is calculated. Record-to-record travel has two types of acceptance criteria. For a minimization problem, if the new solution is less than the record, then the old solution becomes the new solution and the new solution is now the record. Otherwise, if the cost of the new solution is less than the record plus the record times the

deviation [R + (R*D)], then the old solution becomes the new solution and the record is not changed. A new solution is then generated and the process repeats until a stopping condition is met. The best solution from all iterations is stored in memory and becomes the final solution when a stopping criteria has been met (Dueck, 1993).

4.4 Motivation to Employ the Heuristics

Tabu search has been utilized to solve a wide variety of research problems (Glover and Laguna, 1997), while to the best of our knowledge we are aware of only two papers that test the GDA and RTR. Dueck (1993) found GDA and RTR to be superior to simulated annealing for the traveling salesman problem and the problem of the construction of error-correcting codes. Sinclair (1993) compared simulated annealing, genetic algorithms, tabu search, the GDA and RTR to the hydraulic turbine runner balancing problem. Sinclair's results showed that on a balance of ease of implementation, solution quality and solution times, the GDA and RTR performed most satisfactorily while tabu search provided the best solutions, but at the cost of long computation times.

4.5 Parameter Experiment and Selection

The length of the computation time and the quality of the solutions generated by the 3 heuristic algorithms depends primarily on the following key parameters. The length of the tabu list (T_l) and the size of the neighborhood created (N_l) for tabu search, the rain speed parameter (UP) for the GDA, the deviation parameter (D) for RTR, and the appropriate stopping criteria for all three heuristics. A multi-level parameter experiment was conducted that tested 4 problem instances from each of the 3 data sets for each of the 3 objective functions. Each problem instance was replicated 20 times. We evaluated the heuristics parameter effect on computation time using the

Kruskal-Wallis test, and the heuristics parameter effect on solution quality using ANOVA and Tukey's multiple pairwise comparison test. All at a significance level of 0.05.

For tabu search the tabu list will be based on the number of stations and the number of tasks using the decrementing list of Scholl and Voβ (1996) as a guide. For the three data sets the tabu lists to be tested are:

Thomopoulos 19: $\lfloor (K + I)/5 \rfloor, \lfloor (K + I)/4 \rfloor, \lfloor (K + I)/3 \rfloor \Rightarrow (4, 5, 7)$

Kim 61: $\lfloor (K + I)/12 \rfloor, \lfloor (K + I)/10 \rfloor, \lfloor (K + I)/8 \rfloor \Rightarrow (7, 9, 12)$

Arcus 111: $\lfloor (K + I)/15 \rfloor, \lfloor (K + I)/12 \rfloor, \lfloor (K + I)/9 \rfloor \Rightarrow (8, 10, 16)$

where K = # of stations and I = # of Tasks

We generated neighborhoods by randomly selecting one task and creating a neighborhood of a various sizes using $\lceil I/K \rceil$ as a guideline. For the three data sets the neighborhood sizes to be tested are:

Thomopoulos 19: $(\lceil (I/K)/\rceil -1), \lceil (I/K) \rceil, (\lceil (I/K) \rceil +1) \Rightarrow (4, 5, 6)$

Kim 61: $(\lceil (I/K)/\rceil -2), \lceil (I/K) \rceil, (\lceil (I/K) \rceil +2) \Rightarrow (4, 6, 8)$

Arcus 111: $(\lceil (I/K)/\rceil -2), \lceil (I/K) \rceil, (\lceil (I/K) \rceil +2) \Rightarrow (5, 7, 9)$

We also selected a fourth neighborhood size based on the simple formula I-1 where I is the number of tasks. This fourth method allows us to create a neighborhood where the location of one task is swapped with each of the other tasks.

For the traveling salesperson problems (TSP) tested by Dueck (1993) using the GDA he proposed an UP parameter equal to 0.01. Sinclair (1993) also uses an UP parameter equal to 0.01. Since the 3 problem sets we use in our experiments have a wide range of initial objective function values we tested the UP parameter at the following three levels: 0.01, 0.05, 0.10.

For his tests of the TSP using RTR, Dueck (1993) does not provide information on the selection of the deviation parameter (D). Sinclair (1993) uses a D equal to two times the "best_value" (Dueck's Record). This gives an acceptance function of E < 3*"best_value". We conducted small pilot runs for all three data sets and decided to test the following deviation parameters: 0.1, 0.3, 0.5 for the Thomopolous problems; and 0.005, 0.01, 0.05 for the Kim and Arcus problems.

For the GDA and RTR algorithms we tested two stopping criteria to terminate the search procedure: the maximum number of exchanges (N) without accepting a new best solution; and the maximum number of all exchanges (X). We used the following stopping criteria: {(N = 160*I, X = 260*I), (N = 90*I, X = 150*I), (N = 60*I, X = 100*I)}. For tabu search we followed Sinclair (1993) and used the total number of iterations X as the sole stopping criteria (N=X) and set X such that the computation time for tabu search was close to the computation times for the GDA and RTR. Based on a series of small experiments, we selected tabu search stopping criteria of 300 iterations for the Thomopolous problem instances, 400 iterations for the Kim problem instances and 600 iterations for the Arcus problem instances.

For tabu search the length of the tabu list did not have a significant effect on computation time or solution quality, while the neighborhood size had a significant effect on computation time and solution quality. Across all problem instances and objective functions the largest neighborhood size (I -1) had the longest computation time and the lowest objective function values. Using Tukey's test we compared the 3 tabu lists lengths against the maximum neighborhood size and found that the middle tabu list was the most robust across all objective functions and problem sizes. In our empirical research the neighborhood size was based on the

formula I-1 and the tabu list length was set at 5, 9 and 10 for the Thomopolous, Kim and Arcus problems respectively.

For the GDA parameter UP there were no statistically significant differences between UP and computation time, while the results were mixed for the effect of the UP parameter on solution quality. For our stopping criteria we found that for every simulation run the algorithm stopped only after it had reached the maximum number of iterations (X) and that the highest X significantly impacted both computation time and solution quality. The higher the X the longer the computation time and the better the solution quality. In order to determine the impact of UP on solution quality we controlled for problem size and used Tukey's test to compare UP against the highest X. The results were mixed and not statistically conclusive. However, across all 3 objective functions, UP = 0.01 was the most robust for the Thomopolous problems while UP = 0.10 was the most robust parameter for the Kim and Arcus problems, and therefore these parameters were used in the empirical experiments.

Our parameter experiment results for RTR were similar to those for GDA. The deviation parameter (D) does not have a statistically significant effect on computation time or solution quality. For our stopping criteria there were some RTR simulation runs that reached N without accepting a new best solution and therefore terminated the algorithm before X was reached. However, the stopping criteria had a significant effect on computation time and on solution quality. The higher the X the longer the computation time and the better the solution quality. We used the Tukey's test to compare D with the highest X and controlled for problem size in order to determine the impact of D on solution quality. The results were not statistically conclusive. However, across all 3 objective functions, D = 0.3 was the most robust for the Thomopolous

problems while D = 0.01 was the most robust parameter for the Kim and Arcus problems, and therefore these parameters were used in the empirical experiments.

For the stopping criteria of our empirical experiments we dropped the maximum number of exchanges without accepting a new best solution (N) from our algorithms. Stopping criteria was based on the maximum number of all exchanges (X). We dropped N in order to ensure that there would be consistency in the number of possible solutions evaluated by each heuristic. Significantly more computing power became available and we increased the number of iterations for all 3 heuristics. For the GDA and RTR X was set at 10,000 for the Thomopolous problems, 30,000 for the Kim problems and 60,000 for the Arcus problems. For tabu search X was set at 600 for the Thomopolous and Kim problems and 650 for the Arcus problems. These stopping criteria generated similar computation times for all 3 heuristics of approximately 2.7 seconds for the Thomopolous problems, 7 seconds for the Kim problems, and 28 seconds for the Arcus problems.

5 Empirical Results

The three data sets presented in Table 1 were used to generate a variety of problem instances based on the cycle time and the minimum part set. In our empirical experiment we test the following problem instances: 28 Thomopolous, 24 Kim and 16 Arcus for a total of 68 problem instances. For the Thomopolous problem instances we varied the cycle time from 15 to 20 seconds which fixed the number of workstations at 4 and we used 5 different MPSs (231, 232, 332, 225, 732). For the Kim problem instances the cycle time ranged from 71 to 178 seconds to generate 6 or 12 workstations and we used 3 different MPSs (1211, 2121, 1324). For the Arcus problem instances the cycle time ranged from 8100 to 10,300 seconds to generate 15 or 18 workstations and we used 2 different MPSs (11221, 12312). Parameters of the 68 problem

instances can be obtained by contacting the corresponding author. Each problem instance is minimized by the three heuristics for three objective functions giving a total of 612 experiments, and each experiment is replicated 30 times. Our simulations were conducted using MATLAB Version 6 running on a Gateway laptop with 256 MB RAM and 1.7 GHz processor..

In our empirical experiments of the three heuristics we seek to determine if there is a dominant heuristic (best performer) across and within both the problem size and the objective function minimized. We seek to answer the following three research questions in our empirical comparison. For each of the three objective functions:

1. Is there a dominant heuristic for each of the problem sizes tested?
2. Is there a dominant heuristic for all of the problem sizes tested?

For each of the three heuristic algorithms across the three objective functions:

3. Is there a dominant heuristic for all objective functions minimized?

Our answers to these questions will provide valuable insights into the performance of the three heuristics tested in this research to improve the efficiency of the M-UALBP.

5.1 Heuristics Comparison for Problem Sizes

In the following three sections we answer research question 1 and research question 2 for each of the problem sizes tested for the three objective functions. For our analysis in MINITAB Release 13.31 we use general linear model ANOVA with Tukey's multiple comparison test and descriptive statistics. We test the following hypotheses using the Tukey's test at a significance level of 0.05 to determine if there is a heuristic effect on solution quality:

H_0: The population distribution functions are identical

H_1: At least one of the population distribution functions is different

5.1.1 19-Task Thomopolous Problems

Our overall ANOVA results indicated a heuristics effect at a level of 0.0000 for all three objective functions for all problem instances. Analysis of our Tukey's p-value results across all three objective functions is shown in Table 2. The GDA was significantly better than RTR at a level of 0.05 for 66 of 84 problem instances and significantly better than TS for 81 problem instances. RTR was significantly better than TS for 69 problem instances and TS was significantly better than RTR for 2 problem instances. These results indicate that for the 19-task Thomopolous data set the GDA was the dominant heuristic followed by RTR and then TS.

Table 2 Tukey's Test p-Values for 84 Thomopolous Problem Instances

p = 0.05	ADC	MDC	SCV	Total
GDA statistically better than RTR	25	16	25	**66**
GDA statistically better than TS	28	25	28	**81**
RTR statistically better than TS	23	24	22	**69**
TS statistically better than RTR	2	0	0	**2**

In Table 3 we present the average percent reduction from the starting value for each heuristic-objective function combination for 30 simulations. These results support our Tukey's test results. The GDA had the largest overall average percent reduction for all three objective functions followed by RTR and then TS. Note in Table 3 that none of the three heuristics were able to reduce problem instances T23-16a and T23-16b for the MDC objective function. The cycle time of 16 seconds caused the U-line to be highly constrained, with no idle time available.

Insert Table 3 here

5.1.2 61-Task Kim Problems

Our overall ANOVA results again indicated a heuristics effect at a level of 0.0000 for all three objective functions for all problem instances and our Tukey's p-value results at a level of 0.05 are shown in Table 4. For the Kim problems, the GDA was significantly better than RTR for 66

of 72 problem instances and significantly better than TS for 71 problem instances. RTR was significantly better than TS for only 6 problem instances and TS was significantly better than RTR for 61 problem instances. Interestingly, while the GDA was again the dominant heuristic, TS was clearly dominant over RTR.

Table 4 Tukey's Test p-Values for 72 Kim Problem Instances

p = 0.05	ADC	MDC	SCV	Total
GDA statistically better than RTR	24	18	24	66
GDA statistically better than TS	24	24	23	71
RTR statistically better than TS	0	6	0	6
TS statistically better than RTR	24	13	24	61

The average percent reduction from the starting value for the Kim problem instances is shown in Table 5. As for the Thomopolous problem instances, these results also support the conclusions from the Tukey's test. Again, the GDA had the largest overall average percent reduction for all three objective functions, but is now followed by TS and then RTR. Though the overall MDC reductions for TS and RTR were close (24.7% and 21.6% respectively), TS did have a greater reduction for 16 of the 24 problems, of which 13 were significant.

Insert Table 5 Here

5.1.3 111-Task Arcus Problems

For the Arcus problem instances ANOVA results indicated a heuristics effect at a level of 0.0000 for all three objective functions. Table 6 shows Tukey's p-value results for the Arcus problem instances. Once more, the GDA was significantly better than both RTR and TS, while again TS was significantly better than RTR.

Table 6 Tukey's Test p-Values for 48 Arcus Problem Instances

p = 0.05	ADC	MDC	SCV	Total
GDA statistically better than RTR	16	14	16	46
GDA statistically better than TS	16	10	16	42
RTR statistically better than TS	0	0	0	0
TS statistically better than RTR	16	12	16	44

In Table 7 we present the average percent reduction from the starting value for the Arcus problem instances. The GDA had the largest overall average percent reduction for all three objective functions followed by TS and then RTR. None of the three heuristics were able to reduce problem instances A14-8100a and A14-8100b for the MDC objective function. We cannot explain this anomaly since the initial MDC was 1983 seconds and the theoretical lower bound was 28 seconds. In addition, this U-line had 18 work stations. As a comparison, problem instance A14-9700a&b had a theoretical lower bound of only 14 seconds, a higher initial MDC at 2293 and only 15 work stations.

Insert Table 7 Here

5.2 Heuristics Comparison Summary

In sections 5.1.1 to 5.1.3 our answers to research questions 1 and 2 answer our 3rd research question "Is there a dominant heuristic for all objective functions minimized?". It was clear that the GDA dominates both RTR and TS for all three problem sizes and for all three objective functions minimized in this study. Table 8 summarizes our Tukey's results across all heuristics and objective functions and is based on a total of 200 problem instances. In our analysis we drop the four problem instances where none of the three heuristics were able to minimize the initial MDC. Those problems instances being T23-16a&b and A14-8100a&b. From Table 8 it is clear that at a significance level of 0.05, the GDA is significantly better than RTR for 89.0% of the problem instances and significantly better than TS for 97.0% of the problem instances. Our next best performer across all three objective functions is TS since it is a better performer than RTR for both the ADC and SCV objective functions. Tabu search performed better than RTR for the larger problem sizes of 61 and 111 tasks, while RTR performed better for the smaller 19 task

problem set. Table 8 also summarizes the number of times each heuristic found the minimum objective function value for all three heuristics. The GDA generates the minimum objective function value in 196 of 200 problem instances versus 32 and 23 problem instances for RTR and TS, respectively.

Table 8 Tukey's Test Summary Across Objective Functions (n=200)

	ADC	**MDC**	**SCV**	**Total**	**(%)**
Tukey's Results (p = 0.05)					
GDA statistically better than RTR	65	48	65	178	**89.0**
GDA statistically better than TS	68	59	67	194	**97.0**
RTR statistically better than TS	23	30	22	75	**37.5**
TS statistically better than RTR	42	25	40	107	**53.5**
*Number of times the heuristic found the minimum objective function value.					
GDA	67	61	68	196	**98.0**
RTR	7	23	2	32	**16.0**
TS	5	16	2	23	**11.5**

* Percentages do not add up to 100% due to ties.

6 Conclusions and Future Research

For our empirical comparison experiment our overall conclusion is that the great deluge algorithm (GDA) is clearly the best performing heuristic for all problem sizes and objective functions for the M-UALBP. This statement is supported by the Tukey's test, average percent reductions, and the number of times the GDA found the minimum objective function value. For the 19-task Thomopolous problem instances RTR was second best while TS was the second best performer for the larger 61-task Kim and 111-task Arcus problem instances.

This research makes several contributions to the literature. To our knowledge this is the first implementation of the tabu search, the great deluge algorithm and the record-to-record travel algorithm heuristics to solve the M-UALBP. This is also only the fourth study we are aware of that empirically tests a solution for the M-UALBP. Our research has shown that the great deluge algorithm is a robust heuristic for solving the M-UALBP.

The assumptions in our study had some limitations. Travel times for an operator between tasks within a workstation were not included in the initial balance. This is not a critical issue in our research since we only allow a single exchange of tasks. But if multiple tasks can be swapped during an exchange then the size of the workstations might change and hence the travel time. The assumption that task locations are not fixed assumes that tasks can easily be exchanged between workstations at no cost. This assumption would hold for the labor intensive assembly of items such as hand tools or kitchen countertop home appliances where the movement of a task would entail moving the parts bin and the hand-held tools for that task. A second limitation of this study is the use of a random method to exchange tasks for tabu search. Tabu search is based on the idea of using an intelligent mechanism to search the solution space (Glover and Laguna, 1997) and this research used a probabilistic search procedure. Some other limitations of our study are the use of static objective functions, closed workstations (no buffering by workers) and the cost of work-in-process is not accounted for.

For the M-UALBP the following future research is proposed. In conjunction with the either tabu search or the great deluge algorithm, utilize an intelligent search mechanism to identify task exchanges. Scholl and Voβ (1996) use a critical station method and Sparling and Miltenburg (1998) demonstrate a task time correlation coefficient method. The objective functions used in this research were static. Future research could simulate the layouts generated by the objective functions to determine the efficiency of the line, identify bottlenecks and measure inventory levels. Finally an empirical comparison study of mixed-model serial assembly line layouts and U-shaped assembly line layouts could be conducted.

For practitioners the implications of these research are that mixed-model U-shaped assembly lines can be improved through the use of the heuristics utilized in this study, particularly the

easier to implement GDA. This research also found that the solution is sequence dependent and therefore an effort should be made to identify the best sequence.

Acknowledgements

The authors wish to thank Dr. Yeo Keun Kim and Dr. Jae Yun Kim of Chonnam National University for e-mailing us the Kim and Arcus data sets (see Kim, 2002 for a web link). We also wish to thank Dr. Armin Scholl of Friedrich Schiller University Jena and Dr. Robert Klein of Darmstadt University of Technology for providing the ULINO software.

References

Aase, G., Schniederjans, M. and Olson, J. (2003) 'U-OPT: An analysis of exact u-shaped line balancing procedures', *International Journal of Production Research*, Vol. 41, No. 17, pp. 4185-4210.

Aase, G., Olson, J. and Schniederjans, M. (2004) 'U-Shaped assembly line layouts and their impact on labor productivity: An experimental study', *European Journal of Operational Research*, Vol. 156, No. 3, pp. 698-711.

Ajenblit, D. and Wainwright, R. (1998) 'Applying genetic algorithms to the u-shaped assembly line balancing problem', *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 96-101.

Arcus, A. (1963) *An Analysis of a Computer Method of Sequencing Assembly Line Operations*. Ph.D. Dissertation. University of California, Berkley, United States.

Bard, J., Dar-El, E. and Shtub, A. (1992) 'An analytic framework for sequencing mixed-model assembly lines', *International Journal of Production Research*, Vol. 30, No. 1, pp.35-48.

Bard, J., Shtub, A. and Joshi, S. (1994) 'Sequencing mixed-model assembly lines to level parts usage and minimize line length', *International Journal of Production Research*, Vol. 32, No. 10, pp. 2431-2454.

Baykasoğlu, A. (2006) 'Multi-rule multi-objective simulated annealing algorithm for straight and u type assembly line balancing problems', *Journal of Intelligent Manufacturing*, Vol. 17, No. 2, pp. 217-232.

Bukchin, J. (1998) 'A comparative study of performance measures for throughput of mixed-model assembly line balancing in JIT environment', *International Journal of Production Research*, Vol. 36, No. 10, pp. 2669-2685.

Celano, G., Costa, A., Fichera, S. and Perrone, G. (2004) 'Human factor policy testing in the sequencing of manual mixed model assembly lines', *Computers and Operations Research,* Vol. 31, No. 1, pp. 39-59.

Chand, S. and Zeng, T. (2001) 'A comparison of u-line and straight-line performances under stochastic task times', *Manufacturing & Service Operations Management*, Vol. 3, No. 2, pp. 138-150.

Cheng, C., Miltenburg, J. and Motwani, J. (2000) 'The effect of straight and u-shaped lines on quality', *IEEE Transactions on Engineering Management*, Vol. 47, No. 3, pp. 321-333.

Chiang, W., Kouvelis, P. and Urban, T. (2007) 'Line balancing in a just-in-time production environment: Balancing multiple u-lines', *IIE Transactions*, Vol. 39, No. 4, pp. 347-359.

Dar-El, E. and Cother, R. (1975) 'Assembly line sequencing for model mix', *International Journal of Production Research*, Vol. 13, No. 5, pp. 463-477.

Dueck, G. (1993) 'New optimization heuristics: The great deluge algorithm and the record-to-record travel', *Journal of Computational Physics*, Vol. 104, No. 1, pp. 86-92.

Dueck, G. and Scheuer, T. (1990) 'Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing', *Journal of Computational Physics*, Vol. 90, No. 1, pp. 161-175.

Erel, E., Sabuncuoglu, I. and Aksu, B. (2001) 'Balancing of u-type assembly systems using simulated annealing', *International Journal of Production Research*, Vol. 39, No. 13, pp. 3003-3015.

Ghosh, S. and Gagnon, R. (1989) 'A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems', *International Journal of Production Research*, Vol. 27, No. 4, pp. 637-670.

Glover, F. (1977) 'Heuristic for integer programming using surrogate constraints', *Decision Sciences*, Vol. 8, No. 1, pp. 156-166.

Glover, F. and Laguna, M. (1997) *Tabu Search*, Boston: Kluwer Academic Publishers.

Gökcen, H. and Ağpak, K. (2006) 'A goal programming approach to simple u-line balancing problem', *European Journal of Operational Research*, Vol. 171, No. 2, pp. 577-585.

Gökcen, H., Ağpak, K. Gencer, C. and Kizilkaya, E. (2005) 'A shortest route formulation of simple u-type assembly line balancing problem," *Applied Mathematical Modelling*, Vol. 29, No. 4, pp. 373-380.

Hall, R. (1983) *Zero Inventories*, Homewood, IL: Dow Jones-Irwin.

Kim, Y. (2002) 'A set of data for the integration of balancing and sequencing in mixed-model u-lines', http://syslab.chonnam.ac.kr/links/data-mmulbs.doc.

Kim, Y., Kim, S. and Kim, J. (2000) 'Balancing and sequencing mixed-model u-lines with a co-evolutionary algorithm', *Production Planning and Control*, Vol. 11, No. 8, pp. 754-764.

Kim, Y., Kim, J. and Kim, Y. (2006) 'An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model u-lines', *European Journal of Operational Research*, Vol. 168, No. 3, pp. 838-852.

McCormick, S., Pinedo, M., Shenker, S. and Wolf, B. (1989) 'Sequencing in an assembly line with blocking to minimize cycle time', *Operations Research*, Vol. 37, No. 6, pp. 925-935.

Miltenburg, J. (1998) 'Balancing u-lines in a multiple u-line facility', *European Journal of Operational Research*, Vol. 109, No. 1, pp. 1-23.

Miltenburg, J. (2000) 'The effect of breakdowns on u-shaped production lines', *International Journal of Production Research*, Vol. 38, No. 2, pp. 353-364.

Miltenburg, J. (2001a) 'U-shaped production lines: A review of theory and practice', *International Journal of Production Economics*, Vol. 70, No. 3, pp. 201-214.

Miltenburg, J. (2001b) 'One-piece flow manufacturing on u-shaped production lines: A tutorial', *IIE Transactions*, Vol. 33, No. 4, pp. 303-321.

Miltenburg, J and Wijngaard, J. (1994) 'The u-line balancing problem', *Management Science*, Vol. 40, No. 10, pp. 1378-1388.

Monden, Y. (1998) *Toyota Production System: An Integrated Approach to Just-In-Time*, 3rd Edition, Norcross, Georgia: Engineering & Management Press.

Nakade, K. and Ohno, K. (1995) 'Reversibility and dependence in a u-shaped production line', *Queuing Systems*, Vol. 21, Nos. 1-2, pp. 183-197.

Nakade, K. and Ohno, K. (1997) 'Stochastic analysis of a u-shaped production line with multiple workers', *Computers and Industrial Engineering*, Vol. 33, Nos. 3-4, pp. 809-812.

Nakade, K. and Ohno, K. (1999) 'An optimal worker allocation problem for a u-shaped production line', *International Journal of Production Economics*, Vols. 60-61, pp. 353-358.

Nakade, K. and Ohno, K. (2003) 'Separate and carousel type allocations of workers in a u-shaped production line', *European Journal of Operational Research*, Vol. 145, No. 2, pp. 403-424.

Nakade, K., Ohno, K. and Shanthikumar, J. (1997) 'Bounds and approximations for cycle times of a u-shaped production line', *Operations Research Letters*, Vol. 21, No. 4, pp. 191-200.

Ohno, K. and Nakade, K. (1997) 'Analysis and optimization of a u-shaped production line', *Journal of the Operations Research Society of Japan*, Vol. 40, No. 1, pp. 90-104.

Scholl, A. (1999) *Balancing and Sequencing of Assembly Lines*, 2nd Edition, Heidelberg: Physica-Verlag.

Scholl, A. and Klein, R. (1999) 'ULINO: Optimally balancing u-shaped JIT assembly lines', *International Journal of Production Research*, Vol. 37, No. 4, pp. 721-736.

Scholl, A. and Voβ, S. (1996) 'Simple assembly line balancing – heuristic approaches', *Journal of Heuristics*, Vol. 2 No. 3, pp. 217-240.

Schonberger, R. (1982) *Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity*, New York: Free Press.

Simatupang, T. and Sridharan, R. (2002) 'The collaborative supply chain', *The International Journal of Logistics Management*, Vol. 13, No. 1, pp. 15-30.

Simchi-Levi, D., Kaminsky, P. and Simchi-Levi, E. (2000) *Designing and Managing the Supply Chain*, USA: McGraw Hill.

Sinclair, M. (1993) 'Comparison of the performance of modern heuristics for combinatorial optimization on real data', *Computers and Operational Research*, Vol. 20, No. 7, pp. 687-695.

Sparling, D. (1998) 'Balancing just-in-time production units: The *N* u-line balancing problem', *INFOR*, Vol. 36, No. 4, pp. 215-237.

Sparling, D. and Miltenburg, J. (1998) 'The mixed-model u-line balancing problem', *International Journal of Production Research*, Vol. 36, No. 2, pp. 485-501.

Thomopoulos, N. (1967) 'Line balancing sequence for mixed model assembly', *Management Science*, Vol. 14, No. 2, pp. 59-75.

Thomopoulos, N. (1970) 'Mixed model line balancing with smoothed station assignments', *Management Science*, Vol. 14, No. 2, pp. 593-603.

Urban, T. (1998) 'Note: Optimal balancing of u-shaped assembly lines', *Management Science*, Vol. 44, No. 5, pp. 738-741.

Wantuck, K. (1989) *Just In Time for America*, Key Largo, Florida: KWA Media.

Table 3 Thomopolous Problems: Average Percent Reduction for Objective Function and Heuristic

Problem Instance	ADC			MDC			SCV		
	GDA	RTR	TS	GDA	RTR	TS	GDA	RTR	TS
T14-15a	**51.7**	37.6	35.2	**65.0**	58.1	39.7	**65.1**	50.0	44.3
T14-15b	**43.8**	30.6	31.0	**61.7**	51.9	38.6	**55.6**	40.0	39.3
T14-17a	**48.7**	44.2	36.5	**62.5**	48.1	30.3	**96.9**	86.5	71.3
T14-17b	**45.7**	40.4	32.8	**52.4**	45.8	26.1	**92.5**	84.4	68.5
T14-19a	**33.3**	27.4	19.4	40.6	**41.1**	22.2	**95.0**	78.1	55.2
T14-19b	**34.7**	29.4	20.2	39.6	**41.9**	25.9	**92.9**	78.8	54.1
T18-15a	**29.2**	18.8	23.0	**22.1**	20.4	5.4	**30.5**	20.2	21.6
T18-15b	**29.6**	13.4	18.1	**20.8**	16.7	2.9	**30.2**	11.3	15.7
T18-17a	**42.1**	34.7	31.5	**33.3**	24.4	10.0	**85.8**	71.2	64.8
T18-17b	**48.5**	43.1	37.5	**46.1**	40.6	27.5	**87.1**	76.5	61.3
T18-19a	**28.2**	20.1	16.8	7.5	**8.1**	3.5	**66.7**	53.8	44.1
T18-19b	**20.8**	16.2	12.2	6.3	**8.3**	4.0	**48.9**	38.1	29.0
T20-15a	**55.3**	46.5	34.8	**50.0**	27.9	13.8	**60.4**	52.0	39.1
T20-15b	**51.2**	40.6	33.1	**59.3**	39.0	26.0	**56.7**	45.8	37.7
T20-17a	**52.9**	46.7	42.1	**51.2**	42.7	33.6	**95.9**	84.9	75.1
T20-17b	**46.5**	10.9	32.6	**50.9**	40.6	32.1	62.2	**79.2**	64.2
T20-19a	**26.7**	18.5	7.5	6.3	**6.9**	1.9	**81.0**	59.0	23.1
T20-19b	**22.6**	18.3	8.8	**6.3**	**6.3**	2.9	**67.8**	55.1	26.6
T23-16a	**27.0**	22.0	20.5	*0.0*	*0.0*	*0.0*	**28.6**	22.8	20.5
T23-16b	**31.2**	24.2	21.1	*0.0*	*0.0*	*0.0*	**32.0**	22.8	21.0
T23-18a	**59.8**	55.2	49.1	**44.0**	30.0	18.0	**93.5**	86.5	73.9
T23-18b	**62.2**	59.0	54.9	**46.3**	33.0	18.3	**92.8**	88.4	82.7
T23-20a	**36.7**	34.4	22.1	**50.5**	47.5	32.3	**98.7**	95.4	59.5
T23-20b	**44.1**	41.4	27.8	**51.8**	48.4	34.6	**98.3**	94.8	60.1
T24-15a	**60.3**	43.4	27.8	**51.3**	39.3	8.0	**81.3**	63.3	44.1
T24-15b	**59.9**	47.4	36.1	**46.7**	35.3	10.3	**76.7**	63.2	49.3
T24-17a	**40.1**	36.9	31.7	**27.1**	17.5	2.5	**95.5**	93.2	66.0
T24-17b	**31.2**	25.3	21.1	**51.8**	45.9	37.5	**91.1**	75.4	63.5
Avg. Reduction	**41.6**	33.1	28.0	**37.6**	30.9	18.1	**73.6**	63.2	49.1

Bold indicates best heuristic for an objective function for a problem instance

Table 5 Kim Problems: Average Percent Reduction for Objective Function and Heuristic

Problem Instance	ADC			MDC			SCV		
	GDA	RTR	TS	GDA	RTR	TS	GDA	RTR	TS
K3-142a	**59.9**	29.6	53.5	**64.5**	33.5	45.5	**57.9**	28.1	52.9
K3-142b	**49.4**	16.8	36.4	**61.9**	34.2	39.0	**48.8**	17.7	39.0
K3-170a	**31.7**	11.3	25.0	**11.8**	**11.8**	8.9	**68.8**	23.9	55.0
K3-170b	**31.8**	12.7	23.6	11.6	**11.8**	9.2	**64.4**	27.4	49.9
K3-71a	**37.1**	0.6	26.1	**51.3**	21.2	28.3	**37.9**	1.3	25.7
K3-71b	**37.8**	2.3	26.7	**47.5**	21.3	25.5	**38.1**	2.1	26.1
K3-77a	**50.6**	10.3	37.1	**50.2**	31.4	24.0	**68.4**	17.5	49.2
K3-77b	**44.3**	7.9	34.3	**48.4**	29.4	21.9	**62.9**	11.9	48.3
K7-147a	**46.5**	10.4	37.8	**43.8**	11.8	24.2	**46.5**	8.5	42.8
K7-147b	**53.3**	18.0	45.7	**42.3**	5.8	21.4	**56.2**	16.8	44.1
K7-175a	**26.2**	7.3	20.8	7.9	**8.6**	2.0	**62.7**	17.0	46.8
K7-175b	**25.5**	4.9	22.5	7.2	**8.4**	2.8	**63.4**	13.2	47.9
K7-74a	**33.1**	0.5	23.1	**36.1**	7.4	24.4	**35.9**	0.8	24.0
K7-74b	**35.6**	0.6	25.3	**37.4**	11.1	23.8	**37.5**	0.9	22.3
K7-78a	**50.3**	15.3	39.4	**54.2**	31.1	32.5	**68.0**	20.1	56.9
K7-78b	**53.3**	17.1	45.5	**55.0**	32.9	34.3	**71.1**	24.9	58.7
K33-150a	**49.5**	16.3	39.7	**57.1**	31.0	38.5	**48.3**	14.5	40.1
K33-150b	**42.6**	7.7	34.8	**50.5**	20.1	34.8	**42.4**	7.2	35.2
K33-178a	**24.2**	8.5	19.8	19.3	**19.4**	6.5	**63.2**	22.9	51.6
K33-178b	**26.4**	7.0	22.3	19.1	**19.9**	7.4	**66.1**	21.5	56.6
K33-75a	**43.2**	4.2	31.1	**51.8**	28.6	30.8	**43.4**	5.4	36.2
K33-75b	**42.9**	5.8	33.2	**51.3**	26.6	31.3	**43.1**	5.2	35.9
K33-81a	**45.3**	13.9	37.3	**54.3**	30.1	38.0	**69.6**	17.5	56.7
K33-81b	**50.2**	17.6	39.8	**54.3**	31.6	36.6	**73.6**	25.1	62.1
Avg. Reduction	**41.3**	10.3	32.5	**41.2**	21.6	24.7	**55.8**	14.6	44.3

Bold indicates best heuristic for an objective function for a problem instance

Table 7 Arcus Problems: Average Percent Reduction for Objective Function and Heuristic

Problem Instance	ADC			MDC			SCV		
	GDA	RTR	TS	GDA	RTR	TS	GDA	RTR	TS
A14-9700a	**10.0**	0.0	4.9	**36.5**	0.0	29.4	**11.1**	0.1	6.1
A14-9700b	**13.3**	0.0	7.3	**36.3**	0.5	25.9	**16.2**	0.0	7.5
A14-10300a	**9.5**	0.0	5.6	**75.8**	15.3	58.5	**77.8**	0.0	43.4
A14-10300b	**9.1**	0.0	4.6	**74.6**	12.2	63.3	**76.0**	0.6	39.9
A14-8100a	**14.4**	0.0	9.7	*0.0*	*0.0*	*0.0*	**18.7**	0.1	8.1
A14-8100b	**17.2**	0.2	8.7	*0.0*	*0.0*	*0.0*	**22.1**	0.0	11.9
A14-8500a	**12.0**	0.0	8.8	**76.0**	15.4	75.3	**76.4**	0.4	58.5
A14-8500b	**12.1**	0.0	8.3	**77.9**	20.1	74.9	**74.3**	0.1	53.9
A22-9800a	**16.6**	0.2	6.1	**18.6**	2.1	13.4	**31.7**	0.0	16.7
A22-9800b	**18.1**	0.2	10.8	**31.3**	0.6	26.8	**33.9**	0.5	17.5
A22-10300a	**11.1**	0.0	5.9	**75.8**	16.5	71.4	**73.2**	0.2	35.6
A22-10300b	**11.8**	0.0	7.1	**74.9**	14.3	67.1	**73.0**	0.0	46.5
A22-8125a	**8.9**	0.0	5.1	**18.9**	0.0	1.8	**13.2**	0.1	6.3
A22-8125b	**9.0**	0.0	4.1	**19.8**	0.0	1.8	**12.7**	0.1	5.8
A22-8550a	**36.0**	0.1	29.4	**77.2**	30.3	67.4	**89.6**	0.2	65.7
A22-8550b	**36.0**	0.5	28.3	**77.9**	28.7	70.9	**90.7**	0.4	63.0
Avg. Reduction	**15.3**	0.1	9.7	**48.2**	9.8	40.5	**49.4**	0.2	30.4

Bold indicates best heuristic for an objective function for a problem instance