(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 December 2012 (06.12.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/162726 A1

(51) International Patent Classification:
G06F 17/50 (2006.01)

(21) International Application Number:
PCT/AU2012/000576

(22) International Filing Date:
23 May 2012 (23.05.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2011902113 31 May 2011 (31.05.2011) AU

(71) Applicant *(for all designated States except US)*: **MONASH UNIVERSITY** [AU/AU]; Wellington Road, Clayton, Victoria 3800 (AU).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **SARVI, Majid** [AU/AU]; 5 Diamond Court, Doncaster East, Victoria 3109 (AU). **MESBAH, Mahmoud** [AU/AU]; C/- Monash University, Wellington Road, Clayton, Victoria 3800 (AU). **TAN, Jefferson** [AU/AU]; 1 Tulip Court, Notting Hill, Victoria 3168 (AU).

(74) Agent: **PHILLIPS ORMONDE FITZPATRICK**; Level 21, 22 & 23, 367 Collins Street, Melbourne, Victoria 3000 (AU).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[Continued on next page]

(54) Title: OPTIMISING TRANSIT PRIORITY IN A TRANSPORT NETWORK

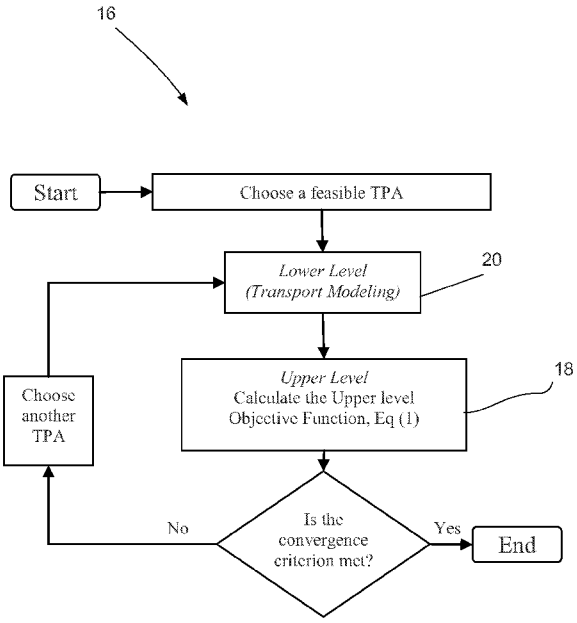

Figure 2

(57) Abstract: A computerised system for optimising transit priority in a network of transport links interconnected by nodes, the system including one or more computing entities including one or more program instruction processing units, and one or more memory devices storing program instructions for performing a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network, the heuristic algorithm including an upper level component for sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and a lower level component for evaluating each TPA using n user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer, wherein the heuristic algorithm uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

WO 2012/162726 A1

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

OPTIMISING TRANSIT PRIORITY IN A TRANSPORT NETWORK

Field of the Invention

The present invention relates generally to methods and systems for allocating space available to vehicles and other entities for movement in a transport network, and in particular to optimising the allocation of exclusive lanes or paths to transit vehicles in such a transport network.

The invention is suitable for use in the allocation of exclusive lanes to transit vehicles such as buses in a road transport network, and it will be convenient to describe the invention is relation to that exemplary, non-limiting application. It will be appreciated however that the invention is also suitable for use in allocating lanes or paths for rail-mounted vehicles, pedestrians, bicycles and other entities in a variety of transport networks.

Background

With an ever increasing travel demand, traffic congestion has become a challenge for many cities around the world. While in many cases, construction of new roads or mass transit is not possible, reallocation of the road space between transit vehicles and cars has emerged as a solution.

Optimization of road space allocation (RSA) from a network perspective can be classified under the umbrella of a Network Design Problem (NDP) (which has a wide range of applications in engineering) in which the goal is to find an optimal combination of links to be added or modified in the network in order to minimize a certain objective function. RSA can be classified as a non-deterministic polynomial-time (NP) hard problem which is computationally challenging and requires the deployment of extensive and advanced computational power to solve. This problem is particularly evident in large scale networks where the number of alternative link combinations to be considered increases exponentially.

Accordingly, there is a need for an efficient method to reduce the number of alternatives to be considered as well as a computational approach to reduce the computer execution time of the analysis. There is also a need for a method of optimising transit priority in a transit network that minimises the computational resources required to be deployed. There is also a need for a method of optimising transit priority in a transit network that ameliorates or overcomes one or more problems, disadvantages or provides an alternative to existing optimising methods.

Summary of the Invention

According to a first aspect of the present invention, there is provided a computerised system for optimising transit priority in a network of transport links interconnected by nodes, the system including:

one or more computing entities including one or more program instruction processing units, and one or more memory devices storing program instructions for performing a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network,

the heuristic algorithm including

an upper level component for sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and

a lower level component for evaluating each TPA using *n* user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer,

wherein the heuristic algorithm uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

In one or more embodiments, a first group of one of more of the computing entities is adapted to process the upper level component of the heuristic algorithm; and

a second group of one or more of the computing entities is adapted to process the lower level component of the heuristic algorithm.

In such embodiments, *p* of the plurality of user behaviour models may be serially evaluated on a same computing entity from the second group, where *p* is an integer less than or equal to *n*.

Alternatively, *p* of the plurality of user behaviour models may be evaluated in parallel on *p* independent program instruction processing units of a same computing entity from the second group, where *p* is the number of processing units on the computing entity and where *p* is an integer less than or equal to *n*.

Yet again, *p* of the plurality of user behaviour models is evaluated in parallel on program instruction processing units of *p* computing entities from the second group, where *p* is an integer less than or equal to *n*.

According to another aspect of the invention, there is provided a computerised method of optimising transit priority in a network of transport links interconnected by nodes in a system including one or more computing entities including one or more program instruction processing units, and one or more memory devices storing

program instructions for performing a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network, the method including:

at an upper level component of the heuristic algorithm, sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and

at a lower level component of the heuristic algorithm, evaluating each TPA using *n* user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer,

wherein the heuristic algorithm uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

According to yet another aspect of the invention, there is provided a first computing entity for use in a computerised system for optimising transit priority in a network of transport links interconnected by nodes, the computing entity including:

one or more program instruction processing units, and one or more memory devices storing program instructions for performing an upper level component of a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network, the upper level component sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and

communication means for receiving data from at least a second computing entity configured to carry out a lower level component of the heuristic algorithm, the lower level component evaluating each TPA using *n* user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer,

wherein the computing entity uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

Brief Description of the Drawings

The invention will now be described in further detail by reference to the accompanying drawings. It is to be understood that the particularity of the drawings does not supersede the generality of the preceding description of the invention.

Figure 1 is a schematic diagram of one embodiment of a computerised system for optimising transit priority in a transport network;

Figure 2 is a flow chart depicting steps involved in a heuristic algorithm carried out by the computerised system of Figure 1;

Figure 3 is a flow chart depicting steps involved in a serial implementation of the heuristic algorithm shown in Figure 2;

Figure 4 is a is a flow chart depicting steps involved in a parallel implementation of the heuristic algorithm shown in Figure 2;

Figure 5 is a schematic diagram of a second embodiment of a computerised system for optimising transit priority in a transport network, adapted to carry out a multi-threading implementation of the parallel heuristic algorithm shown in Figure 4;

Figure 6 is a schematic diagram of a third embodiment of a computerised system for optimising transit priority in a transport network, adapted to carry out a high throughput computing implementation of the parallel heuristic algorithm shown in Figure 4;

Figure 7 is a schematic diagram showing elements forming part of computing entities within the various computerised systems shown in Figures 1, 5 and 6;

Figure 8 is a visual representation of a transport network which the various computerised systems shown in Figures 1, 5 and 6 are adapted to optimising transit priority; and

Figures 9 to 11 are graphs depicting improvements in computation speeds achieved in the above-mentioned embodiments of the computerised system for optimising transit priority in a transport network.

Detailed Description

Referring firstly to Figure 1, there is shown generally a first embodiment of a computerised system 10 for optimising transit priority in a transport network. The system 10 includes a first computing entity 12 and a second computing entity 14. Each computing entity may be a mainframe computer, desktop computer or any programmable machine designed to sequentially and automatically carry out a sequence of arithmetic or logical operations. Each computing entity includes one or more processing units, and one or more memory devices storing program instructions to cause the processing units to execute the program instructions.

The system 10 carries out a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for a transit vehicles in which one or more traffic characteristics are optimised for the network. An exemplary heuristic algorithm 16 is depicted in Figure 2. In this case, the , in this case the heuristic algorithm is a genetic algorithm which mimics the process of natural evolution, however it is to be understood that other forms of heuristic

algorithms may be used in the context of the present invention.

The genetic algorithm includes an upper level component 18, executed by computing entity 12, for sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for transit vehicles. The genetic algorithm also includes a lower level component 20, executed by computing entity 14, for evaluating each TPA using a plurality of user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component. The genetic algorithm 16 uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

Transit Priority Optimisation

The upper level component 18 determines the TPA or the links on which priority would be provided for transit vehicles (decision variables). The function of the upper level is System Optimal (SO), thus the objective function includes a combination of network performance measures. The correspondent constraints are included in the upper level constraints.

The computation carried out by the upper level component 18 can be formulated by the following objective function:

$$MinZ = \alpha \sum_{a \in A} x_a^c t_a^c(x) + \beta \left(\sum_{a \in B} x_a^b t_a^b(x) + \sum_{i \in I} w_i^b \right) + \gamma \sum_{a \in A} \frac{x_a^c}{Occ^c} l_a Imp^c + \eta \sum_{a \in B} f_a s_a Imp^b \quad (1)$$

$$S.t.$$

$$\sum_{a \in A_2'} Exc_a \phi_a \le Bdg \quad (2)$$

$$\phi_a = 0 \text{ or } 1 \quad \forall a \in A_2 \quad (3)$$

It should be understood that $f_a = \sum_{p \in L} f_p \xi_{p,a}$ where $\xi_{p,a}$ is the bus line-link incident matrix and $t_a^b(x)$ is the in-vehicle travel time.

The first two terms in the objective function are the total travel time by car and bus. The next two terms represent the various other impacts of these two modes including emission, noise, accident, and reliability of travel time. The factors $\alpha, \beta, \gamma, \eta$ not only convert the units, but also enable the formulation to attribute different relative weights to the components of the objective function. Equation (2) states that the cost of the implementation should be less than or equal to the budget. The decision variable is ϕ_a by which a system manager operating the computer system 10 tries to minimize the objective function (Z). If $\phi = 1$ buses can speed up to free flow speed, while the capacity of the link for cars is reduced to $Cpc_{1,a}^c$. There are users who determine the link

flows (x). Link flows are related to the decision variables by the formulation of the lower level.

At the lower level, it is users' turn to maximize their benefit. Based on the decision variables determined at the upper level, users make their trips. The computer system 10 implements a four step method for transport modeling. In this embodiment, it is assumed that the travel demand and the distribution of demand are not affected by the location of bus lanes. The lower level component 20 carries out steps to implement three models:

1. Modal split model
2. Traffic assignment model (car demand)
3. Transit assignment model (bus demand)

Once the demand is determined, users choose their travel mode. Then, the car demand segment of the total demand is assigned to the network. The last step at the lower level formulation is the assignment of transit demand. In this embodiment, a known Logit model is used for the mode choice, a known User Equilibrium (UE) model adapted for traffic assignment, and a known frequency based assignment is used for transit assignment. It will be appreciated that many other transport planning models can be used in other embodiments.

The second computing entity 14 is advantageously programmed with a commercially available software system for transportation planning, travel demand modelling and network data management, such as the VISUM package provided by PTV A.G. The use of commercial software in optimization is important as many of the cities are already simulated in such packages. The computer system 10 incorporates the use of this or like packages in the optimization without a need to convert the model to other platforms. The first computing entity 12 can be programmed with any suitable software with mode choice and assignment component for private and public transport.

This bi-level structure, even in the simplest form, with linear objective function and constraints is a NP-hard problem and is difficult to solve. Furthermore, in this problem, the upper level objective function and the UE traffic assignment are non-linear. This adds to the complexity of the problem. However, a heuristic algorithm such as Genetic Algorithm approach is well suited to find an approximate solution to the optimal answer. The output of the model is the combination of transit exclusive lanes which minimizes the proposed objective function.

Genetic Algorithm

In order for the skilled addressee to better comprehend the invention, a Genetic Algorithm (GA) is used here as an example of heuristic algorithm. GA is an iterative search method in which the new answers are produced by combining two predecessor answers. Inspired from evolutionary theory in nature, GA starts with a feasible set of answers referred to as the *population*. Each individual answer in the population (called a chromosome) is assigned a survival probability, based on the value of the objective function. Then, the algorithm *selects* individual chromosomes based on this probability to breed the next generation of the population. GA uses *cross over* and *mutation* operators to breed the next generation which replaces the predecessor generation. The algorithm is repeated with the new generation until a convergence criterion is satisfied.

In the computer system 10, a GA is applied to the RSA problem. A gene is defined to represent the binary variable ϕ and a chromosome is the vector of genes (Φ) which represents a TPA. A chromosome (or TPA) contains a feasible combination of links on which an exclusive lane may be introduced (set A_2). Therefore, the length of the chromosome is equal to the size of A_2. The algorithm starts with a feasible initial population. The chromosomes of the initial population are produced randomly.

Once a feasible chromosome population is produced, the upper level objective function for all chromosomes (or TPAs) should be determined. TPA evaluations are carried out at the lower level by the transport planning models of mode split, traffic assignment, and transit assignment. Using the flow and travel time at the lower level, the objective function for the chromosome is determined. The lower level calculations are repeated for all chromosomes in the population.

The above process can be summarized in the following series of steps:

Step 0: Initialization: Set iteration number (n)=1, best answer value or upper bound (UBD)=∞. Set max generations (maxg), and number of generations with same UBD, m.

Step 1- Initialization: Generate initial population.

Step 2 - Evaluation: Calculate the objective function value for all chromosomes (or TPAs) in the population, using the transport planning models at the lower level.

Step 3 - Fitness: Determine survival probabilities (fitness) and update UBD.

Step 4 - Convergence: If n>maxg or UBD is constant for m generations stop.

Step 5 - Reproduction: Breed new generation by performing selection, cross over, and mutation. Go to Step 2.

Implementation of the Genetic Algorithm

The most computationally intensive part of the GA is step 2 where TPAs are evaluated. The execution time for one evaluation is large. Furthermore, the GA requires large number of TPA evaluations. One evaluation involves running the four-

step modeling for a network which may take as long as some hours on a personal computer. The number of TPA evaluations depends on the number of decision variables and attributes of the GA such as cross over probability and mutation probability. The goal of this section is to decompose the processes of the GA in order to execute them simultaneously on separate resources. Such an approach can significantly reduce the execution time of the GA.

The steps of Genetic algorithm in terms of dependency of processes are of two types: First is the evaluation step (step 2). The evaluation of an individual chromosome (or TPA) is independent of other chromosomes (or TPAs) in a generation. Therefore, step 2 consists of a number of processes that can be analyzed independently. The second part of GA involves fitness, convergence, and reproduction (steps 3 to 5) which should be performed on the population. This part of the GA integrates the individual evaluations of step 2 where the processes are interdependent. On the basis of the dependency attribute two variants of the GA can be implemented, namely a Serial Genetic Algorithm (SGA) and a Parallel Genetic Algorithm (PGA).

In the SGA 22 depicted in Figure 3, all of the processes are carried out in a sequence which means that in step 2, evaluation of a chromosome is performed before evaluation of another chromosome is started. Then steps 3, 4, and 5 are completed to produce another generation and cycle back to step 2. The SGA is the simplest variant of GA to implement as a computer.

However, in the PGA, the independency attribute of the evaluations are utilized to perform the evaluations simultaneously. In the PGA 50 depicted in Figure 5, step 2 is executed in parallel which is then followed by steps 3, 4, and 5 in a sequence. The aim of this parallelization is to achieve faster computation. Two PGA implementation techniques can be employed: using multiple cores available on one machine or multiple cores available on multiple machines in a network, as will be now explained.

Parallel GA – Multi-Threading (MT) Approach

The operating system (OS) of the first computing entity 32 creates process *threads* to run computer applications. If no provision is made in the coding, the OS creates only one thread by which all the required processes of the code are run successively. In the presence of multiple running applications, there could be more than one thread processed by a Central Processing Unit (CPU) at a time. Execution of multiple threads can result in parallel processing of an application if the machine is capable of handling it, such as when a *multi-core* machine is in use. This technique of executing multiple threads in parallel is called *Multi-threading* in computer science.

In order to implement PGA by the Multi-threading technique, the architecture 28 shown in Figure 4 is used. The number of threads is selected equal to the number of processing cores on a machine (say *p*) plus a main thread. The main thread is reserved to control the flow of the GA from the start to the end. The main thread performs the fitness, convergence, and reproductions steps of the GA. The rest *p* threads are used to execute the TPA evaluations (Objective function evaluation). Please note that the main thread is paused when the *p* threads are running so that all cores would be free to execute the remaining p threads. This technique can engage multiple cores available on one computer.

The speed up achieved by the multi-threading approach depends on the number of cores on a machine and the efficiency of the OS on running parallel processes. In this study, the implementation is in the Microsoft Windows OS since the TPA are evaluated by VISUM and this package is only available in the Windows platform. Windows is at times criticized for its performance, but regardless of the operating system, there will be cases where performance starts to decline when the number of threads exceeds hardware cores. This is where the OS must share the limited hardware resources among so many threads that must be supported, due to time slicing overheads.

Moreover, from the hardware point of view, the maximum number of cores that can be installed on one machine is limited by space and temperature requirements of the hardware. There can also be a limit to performance gains due to memory latency and cache conflicts in multithreaded computations. Therefore, for a number of reasons, there is a cap on the maximum achievable speed up using the Multi-threading approach. Finally, one cannot escape the fact that additional cores in a given machine cost additional expenditure.

However, given that the TPA evaluations are carried out using a commercial package, and all commercial packages need licenses, multi-threading approach can save considerably on *license* costs for some packages. Since all instances of a commercial package (e.g. VISUM) are triggered on one machine, only a single VISUM license is sufficient to be provided, but at the expense of performance. It makes sense to strike a balance between budget and performance costs, and the next section discusses how the HTC approach can provide the benefits of parallelism, avoiding some of performance limits of Multi-threading, although it requires multiple licenses of the software. A computer code is developed for both the multi-threaded (MT) and the HTC approach in Visual Basic .NET environment in this study.

Figure 5 depicts a computer system 30 adapted to carry out the MT approach to PGA. The computer system 30 implements a distributed architecture and includes a

first computing entity 32 in communication with a second computing entity 34. The second computing entity 34 includes multiple processing cores 36 to 42, and is programmed with a commercially available software system for transportation planning, travel demand modelling and network data management, such as the VISUM package provided by PTV A.G.

Parallel GA – The HTC Approach

The multi-threading approach can run a certain number of TPA evaluations in parallel. For example in the architecture 28 shown in Figure 4, *p* evaluations from *n* evaluations in a population can be run in parallel. The ideal case is when *p* is equal to *n*, which means all *n* evaluations can be done at the same time. Hence, multi-threading is an approach that delivers speed-ups due to the parallelism afforded by many computations that can run simultaneously on many cores. However, as mentioned earlier, the number of threads on a machine is limited, or at least, there is a limit to the speed up expected. A distributed computing approach such as HTC schedules TPA evaluations as computational jobs to several computers on a network, each being independent with its own set of cores and local memory. Therefore, there is virtually no limit on the number of processing tasks that can be done simultaneously as the number of computers in a network is not necessarily bounded. The trade-off is a need to distribute the task to available computers in the network, manage the queue, data transfers, provide an inter-process message-passing system in some cases, then collect and integrate the results. There are a number of existing systems that provide for these things.

Figure 6 depicts a computer system 60 adapted to carry out the HTC approach to PGA. The computer system 60 implements a distributed architecture and includes a first computing entity 62 in communication with a group of second computing entities 64 to 72 by means of an intermediary server 74.

Once again, the group of second computing entities 64 to 72 are programmed with a commercially available software system for transportation planning, travel demand modelling and network data management, such as the VISUM package provided by PTV A.G. The intermediary server 74 includes software to perform communications, job queuing and job distribution functions in relation to the group of second computing entities 64 to 72, such as the Condor open-source high throughput computing software developed by the Condor team at the University of Wisconsin-Madison.

The various computing entities described above may be implemented using hardware, software or a combination thereof and may be implemented in one or more

computer systems or processing systems. An exemplary computer system 80 is shown in Figure 7. The computer system 80 includes one or more processors 82. The processor(s) 82 is/are connected to a communication infrastructure 84. The computer system 80 may include a display interface 86 that forwards graphics, texts and other data from the communication infrastructure 84 for supply to the display unit 88. The computer system 80 may also include a main memory 90, preferably random access memory, and may also include a secondary memory 92.

The secondary memory 92 may include, for example, a hard disk drive 94, magnetic tape drive, optical disk drive, etc. The removable storage drive 96 reads from and/or writes to a removable storage unit 98 in a well known manner. The removable storage unit 98 represents a floppy disk, magnetic tape, optical disk, etc.

As will be appreciated, the removable storage unit 96 includes a computer usable storage medium having stored therein computer software in a form of a series of instructions to cause the processor(s) 82 to carry out desired functionality. In alternative embodiments, the secondary memory 92 may include other similar means for allowing computer programs or instructions to be loaded into the computer system 80. Such means may include, for example, a removable storage unit 100 and interface 102.

The computer system 80 may also include a communications interface 104. Communications interface 104 allows software and data to be transferred between the computer system 80 and external devices. Examples of communication interface 104 may include a modem, a network interface, a communications port, a PCMIA slot and card etc. Software and data transferred via a communications interface 104 are in the form of signals 106 which may be electromagnetic, electronic, optical or other signals capable of being received by the communications interface 104. The signals are provided to communications interface 104 via a communications path 104 such as a wire or cable, fibre optics, phone line, cellular phone link, radio frequency or other communications channels.

Although in the above described embodiments the invention is implemented primarily using computer software, in other embodiments the invention may be implemented primarily in hardware using, for example, hardware components such as an application specific integrated circuit (ASICs). Implementation of a hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant art. In other embodiments, the invention may be implemented using a combination of both hardware and software.

Adaptation of the RSA problem

Although the issues emerging in adopting a general HPC tool to the RSA problem are tool specific, important lessons can be learned from them for application of any commercial package on the Windows platform. As VISUM is a commercial package, it is protected from being used beyond the maximum number of running instances that is allowed by the purchased license. Additionally, VISUM comes with a hardware lock, which contains the license data. The first requirement was to obtain a network license, installed on a license server to which all computers on the network should have access. The license server can be the same machine as or different to the Condor *central manager*. It should be noted that if *n* computers are in the network and *m* licenses are available, the maximum number of parallel TPA evaluations is *min(n, m)*.

To evaluate a function, a user submits a *job* on the *submission machine*, and Condor will copy the application program to the *worker* node as a job or *task*. When the job is completed, any output data file identified in the job's *command file* for copying back will be downloaded back to the submission machine, and then the application copy is removed on the worker node. However, the size of VISUM (700 MB) requires a considerable amount of time for copying across from one machine to another, even on a typically fast network, e.g., 1000 Mbps. In order to exploit the parallelism on a Condor resource, VISUM would have to be copied to several worker nodes. Furthermore, VISUM has to be installed and registered in Windows in such a way that user interaction with a mouse and keyboard is normally required. The solution was pre-installation of VISUM on a subset of computers in the network. Condor must then be told, in the job command file, to send jobs only to nodes with VISUM installed.

Windows differentiates between the local or remote launch of an application. Windows also consults the user permissions to run an application either locally or remotely. Condor runs VISUM remotely as a 'system' user. The VISUM 'COM server' was set to grant suitable permissions to launch Condor jobs from a 'remote system' user.

Numerical Example

In this section, three exemplary GA implementation approaches (SGA, PGA-MT, and PGA-HTC) are applied to an example network 120 shown in Figure 8. These examples show how the developed method can be applied using a heuristic algorithm such as GA. This grid network consists of 86 nodes and 306 links. All the circumferential nodes together with Centroid 22, 26, 43, 45, 62, and 66 are origin and destination nodes. A 'flat' demand matrix of 30 Person/hr is travelling from all origins to all destinations. The total demand for all the 36 origin destination is 37,800 Person/hr.

There are 10 bus lines covering transit demand in the network shown in Figure 8. The frequency of service for all the bus lines is 10 mins. The models and parameters used in this example are extracted from those calibrated for Melbourne Integrated Transport Model (MITM) a four step transport model used by the Victorian State Government for planning in Melbourne (Department of Infrastructure, 2004).

Vertical and horizontal links are 400m long with two lanes in each direction and the speed limit of 36 km/hr. It is assumed that if an exclusive lane is introduced on a link on one direction, it may not necessarily be introduced in the opposite direction. There are a total number of 120 links (one directional) in the network shown in Figure 8 on which an exclusive lane can be introduced. These links are highlighted in black solid line in Figure 8.

The following Akcelik cost functions are assumed for links with an exclusive lane (Equation (15)) and without exclusive lane (Equation (16)):

$$t_{1,a}^{c} = t_{0,a} + \frac{3600a}{4}[(\frac{x_a^c}{Cap_{1,a}^c} - 1) + \sqrt{(\frac{x_a^c}{Cap_{1,a}^c} - 1)^2 + \frac{8b}{ad}(\frac{x_a^c}{Cap_{1,a}^c})}], t_{1,b}^{t} = t_{0,a} \quad (4)$$

$$t_{0,a}^{c} = t_{0,a}^{b} = t_{0,a} + \frac{3600a}{4}[(\frac{x_a^c + x_a^t}{Cap_{0,a}^c} - 1) + \sqrt{(\frac{x_a^c + x_a^t}{Cap_{0,a}^c} - 1)^2 + \frac{8b}{ad}(\frac{x_a^c + x_a^t}{Cap_{0,a}^c})}] \quad (5)$$

where t_0 determines travel time with free flow speed, a is length of observation period, b is a constant, d is lane capacity, and other terms are as defined earlier. It is assumed that each link has 2 lanes and:

$a = 1hr, b = 1.4, d = 800veh/hr$

$Cap_{0,a}^{c} = 1800veh/hr$

$Cap_{1,a}^{c} = 900veh/hr$

Mode share is determined using a Logit model. Traffic User Equilibrium (UE) and a frequency-based assignment is employed to model traffic and transit assignments, respectively. All these lower level transport models are implemented using Visum modeling package (PTV AG, 2009).

The upper level objective function includes total travel time (veh.sec) and total vehicle distance (veh.km). The absolute value of the objective function therefore can be very large. In order to avoid numerical problems, a constant value is subtracted from the objective function for all evaluations. Hence, the value of the objective functions is on a relative basis. The weighting factors of the objective function are assumed to be equal to 0.01. Regarding constraints, it is assumed that the budget allows for all candidate links for the provision of bus priority.

Genetic Algorithm has many parameters to tune. It was assumed that population size, cross over probability (cp), mutation probability (mp) respectively are

40, 0.98, 0.01. The aim of this example is to demonstrate the computation time saving by the HTC approach in comparison to the serial approach. Although selection of the GA parameters may vary the absolute value of the execution time, the time differences on a relative basis are useful indicators to highlight the efficiency of the HTC approach.

Table 1 below shows the seven computers used in this study. The pool of computers includes various CPUs, Windows versions, and VISUM versions. It demonstrates that the HTC approach can incorporate all types of computers and software versions. When a machine for example has 8 threads, it indicates that it can perform a maximum of 8 TPA evaluations at a time. A total of 32 threads are provided which means if all computers could be assigned to an evaluation job, a total of 32 evaluations can be carried out simultaneously. This requires 32 VISUM licenses. The last column in Table 1 is the time spent for evaluation of one TPA on each machine. As the table shows, machine 1 was the fastest computer with 65 seconds of execution time and machine 7 was the slowest with 226 seconds.

machine	CPU	Cores	Threads	Windows	VISUM	Evaluation Time (s)
1	Intel Core i7 CPU 860 @ 2.8 GHz	4	8	7 64-bit	11.03 64-bit	65
2	Intel Core i7 CPU Q820 @ 1.73 GHz	4	8	7 64-bit	11.03 64-bit	147
3	Intel Core 2 Quad CPU Q6600 @ 2.4 GHz	4	4	7 64-bit	11.03 64-bit	101
4	Intel Core 2 Quad CPU Q6600 @ 2.4 GHz	4	4	XP 64-bit	11.01 32-bit	122
5	Intel Core 2 Quad CPU Q6600 @ 2.4 GHz	4	4	XP 64-bit	11.01 32-bit	121
6	Intel Core 2 Duo CPU E8500 @ 3.16 GHz	2	2	XP 64-bit	11.01 32-bit	88
7	Intel Pentium 4 CPU @ 3.2 GHz	2	2	XP 32-bit	11.01 32-bit	226

Table 1: Computers used in the experiments

Three types of SGA, PGA by Multi-threading (MT) approach, and PGA by HTC approach are run in this section. For the PGA runs, a range of cores/threads could be available based on which the execution time is different. The First experiment (datum) for the MT approach is performed on machine 4 with 4 threads, and for the HTC approach on machines 1, 2, 3, and 6 with a total of 22 threads.

The GA is a heuristic method which at each run may take a different path towards the optimal answer. In this section first it is shown how GA moves towards the optimal answer. Then it will be demonstrated that the parallelization approaches, including MT and HTC, do not affect the number of evaluations in the GA or the rate of improvement in the objective function. The parallelization approaches merely reduce the time required for evaluations by performing a number of evaluations in parallel. The minimum objective function value achieved in a run with 400 generations was equal to -4.757.

Figure 9 presents a graph 130 showing the improvement in the objective function value for four SGA runs where 3 runs were terminated short in generation 50 and the fourth run was continued for 300 generations. As it will be discussed, the execution time of SGA is prohibitively long; thus, the number of generations cannot increase to more than 300 in this example.

All of the runs evaluated about 1700 TPAs by generation 50. The fourth run (SGA-4) is continued and completed a total of 5897 TPA evaluations at the end of generation 300. This figure shows that although these runs do not follow exactly the same path in reducing the objective function value, the objective function value improves gradually by an increase in the number of evaluations.

Figure 10 presents a graph 140 showing the improvement of the objective function for two MT and two HTC runs. For comparison purposes, the graph of SGA runs is also repeated in Figure 10. As visually can be verified, all approaches represent a similar trend in reduction of the objective function. This result is expected as the same algorithms are used for fitness, convergence, and reproduction of the SGA, MT and HTC. Therefore, the algorithm to produce TPAs in all approaches is the same, although the evaluation step (step 2) is implemented differently. The implementation of the evaluation step has resulted in various execution times.

A graph 140 showing the improvement of the objective function by an increment in the execution time is presented in Figure 11. The SGA runs have a significantly longer execution time than the PGA runs. In comparison of MT and HTC which are both PGA, Figure 11 suggests that HTC runs are faster. For example, in order to reduce the objective function value to 30, SGA-3 spends about 170’000 sec (about 2 days) while HTC needs only 2000 sec which is about one-ninth of what it takes for the SGA approach. Please note that the execution time of SGA’s fourth run (SGA-4) with 300 generations exceeded 5 days, which is prohibitively long to run GA as SGA for this example. This overall comparison reveals that parallelization can effectively reduce the GA execution time. A more specific comparison between the implementation approaches is made in Table 2.

Three sets of experiments were organized with population size of 40, cross over probability (cp) of 0.98, and mutation probabilities (mp) of 0.005, 0.01, and 0.02. The change in the mp, can change the number of evaluated TPAs (Mesbah et al., 2011). Table 2 also verifies this observation where the number of evaluations on Generation 50 has increased by an increase in the mp.

The following three performance measures are used in this study:

1. Average Time per Evaluation (ATE) which is the ratio of the execution time to the number of TPA evaluations
2. Speed up which is the ratio of ATE in one run to the ATE associate with a SGA run.
3. Efficiency which is the ratio of the speed up to the number of available threads.

According to the above definitions, the speed up and the efficiency of SGA runs are 1. Table 2 shows that ATE is almost constant with a change in the mp. The ATE for SGA, MT, and HTC runs are approximately 140sec, 40sec, and 14sec, respectively. This shows that the PGA approaches and specifically HTC can reduce the execution time of the GA by more than 10 times. A speed up of more than 3 times is achieved with the application of the MT approach while the speed up was more than 9 times using the HTC approach. The efficiency measure demonstrates that in return for adding each thread in the MT approach, the execution time has improved by 80-90%. However, the efficiency in the HTC approach was just above 50% for addition of each thread. This gap is caused by the fact that there is an overhead time associated with distribution and queuing of the processing jobs in the HTC approach. The speed up and efficiency measures are relatively consistent across the tests with mp=0.005, 0.01, and 0.02.

Experiment Code	Approach	Number of Evaluations on Generation 50	Execution Time (sec)	Average time per evaluation	Number of Cores	Number of Threads	Speed up	Efficiency
E218	SGA	1649	240618	145.9	1	1	1	1
E220	MT	1513	59865	39.6	4	4	3.687	0.922
E219	HTC	1454	21607	14.9	10	18	9.821	0.546
			mp =0.005					
E210	SGA	1680	241475	143.7	1	1	1	1
E223	MT	1543	66480	43.1	4	4	3.335	0.834
E227	HTC	1626	24918	15.3	10	18	9.378	0.521

			mp =0.01					
E215	SGA	1721	231197	134.3	1	1	1	1
E224	MT	1714	72237	42.1	4	4	3.187	0.797
E214	HTC	1683	24204	14.4	10	18	9.343	0.519
			mp =0.02					

Table 2: Comparison of the speed up using MT and HTC approaches

Table 3 below presents the effects of the number of available threads on the execution time in the HTC approach. In all of the experiments/runs population size is 40, cp=0.98, and mp=0.01. In general, the ATE decreases when the number of threads increases which is an expectable outcome. The lowest ATE belonged to experiment E228 with 11.7sec. However, two important observations can also be made from Table 3. First is that the ATE has remained constant when the number of threads increased from 22 to 26. This means although the resources to perform the evaluation job were increased, the ATE was not improved. A closer look at the log files of the experiments reveals the reason. About 1650 TPA evaluations are performed in each run which is an average of 1650/50=33 evaluations per generation. Nevertheless, this is not uniformly distributed. The TPA evaluations are recorded to prevent evaluating a TPA twice. This technique saves repeating evaluations which considerably reduces the GA execution time. Therefore, although the average number of evaluations per generation is 33, the first generations evaluate close to 40 (which is the population size) evaluations, while the last generations evaluate just over 20 TPAs. When close to 40 TPAs are being evaluated, both experiments of E228 and E230 may allocate 2 or less evaluations to a thread. This means about 2 evaluations are being run in a sequence. Similarly, when just above 20 TPAs are being evaluated, both E228 and E230 have enough threads to run all evaluations simultaneously. Therefore, with an increase of just 4 threads, the execution time was not improved.

According to the above logic, the ATE in experiment E231 should also be similar to E228 and E230. However, as mentioned in Table 3, ATE has considerably increased in E231. In this experiment, a very slow CPU of machine 7 is added to the pool. As shown before in Table 1, machine 7 has the slowest CPU. During the time which is required for this machine to evaluate one TPA, other machines can evaluate between 2 to 4 TPAs. As a result, this machine only holds up the other available threads as ‘idle’ which extends the evaluation time of each generation and collectively increases the evaluation time of the experiment.

Exper-iment Code	Number of Cores	Number of Threads	Computers	Number of Evaluations on Generation 50	Execution Time (sec)	Average time per evaluation (sec)	Speed up	Efficiency
E210	1	1	4	1680	241475	143.7	1	1
E225	6	10	2, 6	1724	37296	21.6	6.643	0.664
E226	10	14	2, 4, 6	1481	28013	18.9	7.597	0.543
E227	10	18	1, 2, 6	1626	24918	15.3	9.378	0.521
E228	16	22	1, 2, 4, 6	1659	19335	11.7	12.331	0.560
E230	20	26	1, 2, 4, 5, 6	1614	19053	11.8	12.173	0.468
E231	22	28	1, 2, 4, 5, 6, 7	1645	26235	15.9	9.011	0.322
			mp =0.01					

Table 3: Comparison of the HTC speed up by varying cores

The foregoing describes the development and implementation of High Throughput Computing (HTC) to a transport optimization problem. Road Space Allocation (RSA) optimization problem is tackled using 3 variants of a Genetic Algorithm, namely Serial Genetic Algorithm (SGA), Parallel Genetic Algorithm (PGA) using Multi-threading (MT) technique and PGA using HTC. The final answer of the optimization problem was independent from the type of GA variant.

The performance of the GA variants has been compared in a numerical example. The PGA-MT approach with 4 'threads' available could reduce the execution time by 3.2 to 3.7 times compared to SGA. The PGA-HTC approach with 18 'threads' could decrease the execution time by 9.3-9.8 times in different examples. Although the 'efficiency' of the MT approach was higher than the HTC, MT approach cannot be used to solve large scale (real world network) examples since the total number of threads on a computer is limited. In contrast, there is no limit on the number of cores/threads which can be employed in the HTC approach. One of the novel aspects of this study was the successful implementation of the HTC approach to the road space application study using commercial software on Windows platform.

On the other hand, the overhead of pre-installing commercial applications similar to PTV's VISUM cannot be taken for granted. There is tremendous benefit in how grid computing, yet another approach to HTC, makes it easy for users to reach out to resources that are plugged into the grid by a new or growing member of the virtual organization. On grids, users typically require standard libraries and applications that

are already installed on typical Linux nodes. This is not the case when a commercial package like VISUM is involved. Therefore, a reasonable follow-up to this project could be to explore *cloud computing* (Foster et al., 2008) as a flexible HTC resource. Clouds use *virtual machines*, instances of an operating system that are running, usually with a number of other instances, on a particularly powerful machine. *Virtualization* allows multiple diverse operating environments to run on top of the actual system of the base machine. Clouds can accommodate a user-prepared virtual machine *image* with the preferred settings and applications configured, and run *several* of them in a virtual network. The user therefore, has considerable control over how much computational power is desired in an HTC resource that can be created, expanded, contracted or retired as necessary. The next step in our work is therefore, to explore the applicability, benefits and constraints using cloud environments.

From transportation engineering perspective, the proposed computational method was implemented for a RSA problem. Nevertheless, the proposed framework is generic enough to be applied to the entire family of Network Design Problem (NDP). This study can be expanded by applying the framework to NDP problems in large scale networks. Moreover, substitution and comparison of other heuristic methods with the GA could be another area of future studies.

While the invention has been described in conjunction with a limited number of embodiments, it will be appreciated by those skilled in the art that many alternative, modifications and variations in light of the foregoing description are possible. Accordingly, the present invention is intended to embrace all such alternative, modifications and variations as may fall within the spirit and scope of the invention as disclosed.

Claims

1. A computerised system for optimising transit priority in a network of transport links interconnected by nodes, the system including:

one or more computing entities including one or more program instruction processing units, and one or more memory devices storing program instructions for performing a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network,

the heuristic algorithm including

an upper level component for sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and

a lower level component for evaluating each TPA using *n* user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer,

wherein the heuristic algorithm uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

2. A system according to claim 1, wherein

a first group of one of more of the computing entities is adapted to process the upper level component of the heuristic algorithm; and

a second group of one or more of the computing entities is adapted to process the lower level component of the heuristic algorithm.

3. A system according to claim 2, wherein

p of the plurality of user behaviour models is serially evaluated on a same computing entity from the second group, where *p* is an integer less than or equal to *n*.

4. A system according to claim 2, wherein

p of the plurality of user behaviour models is evaluated in parallel on *p* independent program instruction processing units of a same computing entity from the second group, where *p* is the number of processing units on the computing entity and where *p* is an integer less than or equal to *n*.

5. A system according to claim 2, wherein

p of the plurality of user behaviour models is evaluated in parallel on program instruction processing units of *p* computing entities from the second group, where *p* is an integer less than or equal to *n*.

6. A computerised method of optimising transit priority in a network of transport links interconnected by nodes in a system including one or more computing entities including one or more program instruction processing units, and one or more memory devices storing program instructions for performing a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network, the method including:

at an upper level component of the heuristic algorithm, sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and

at a lower level component of the heuristic algorithm, evaluating each TPA using *n* user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer,

wherein the heuristic algorithm uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

7. A method according to claim 6, and further including

processing the upper level component of the heuristic algorithm at a first group of one of more of the computing entities; and

processing the lower level component of the heuristic algorithm at a second group of one or more of the computing entities.

8. A method according to claim 7, and further including

serially evaluating *p* of the plurality of user behaviour models on a same computing entity from the second group, where *p* is an integer less than or equal to *n*.

9. A method according to claim 7, and further including

evaluating in parallel *p* of the plurality of user behaviour models on *p* independent program instruction processing units of a same computing entity from the second group, where *p* is the number of processing units on the computing entity and where *p* is an integer less than or equal to *n*.

10. A system according to claim 7, wherein

evaluating in parallel *p* of the plurality of user behaviour models on program instruction processing units of *p* computing entities from the second group, where *p* is an integer less than or equal to *n*.

11. A first computing entity for use in a computerised system for optimising transit priority in a network of transport links interconnected by nodes, the computing entity including:

one or more program instruction processing units, and one or more memory devices storing program instructions for performing an upper level component of a heuristic algorithm to determine a Transit Priority Alternative (TPA) defining a combination of transport links on which priority would be provided for transit vehicles in which one or more traffic characteristics are optimised for the network, the upper level component sequentially generating a plurality of TPAs each defining a different combination of transport links on which priority would be provided for a transit vehicles; and

communication means for receiving data from at least a second computing entity configured to carry out a lower level component of the heuristic algorithm, the lower level component evaluating each TPA using *n* user behaviour models to evaluate the one or more traffic characteristics for each TPA generated by the upper level component, where n is an integer,

wherein the computing entity uses the results of the lower level component evaluation of each TPA in the generation of a subsequent TPA.

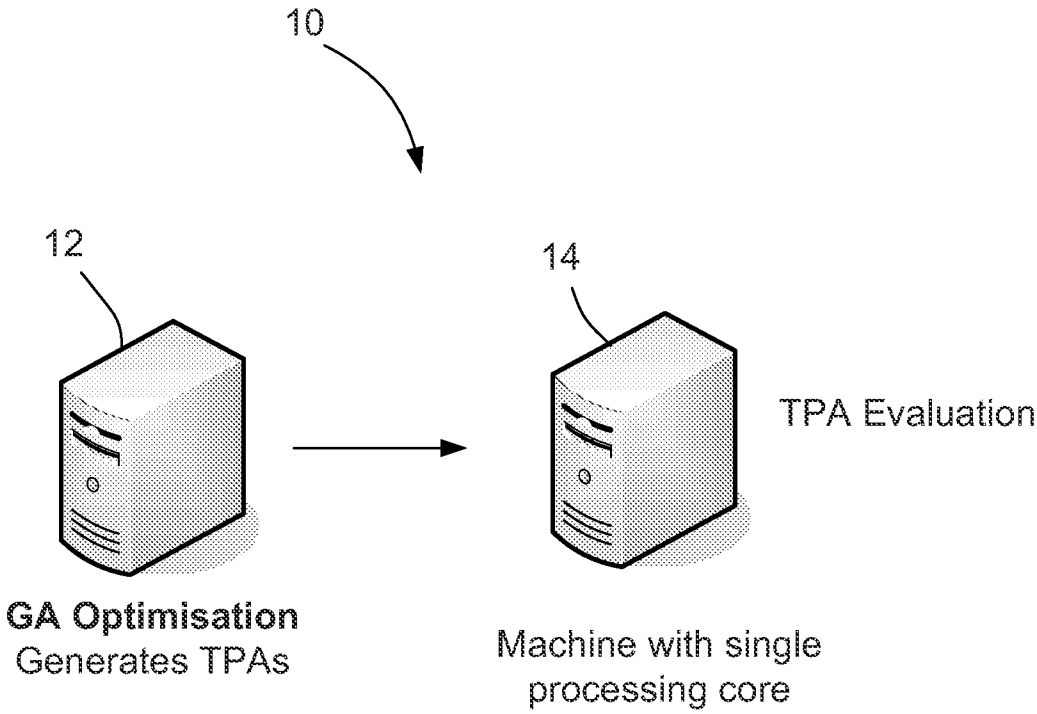
10
12
14
TPA Evaluation
GA Optimisation
Generates TPAs
Machine with single
processing core

Figure 1

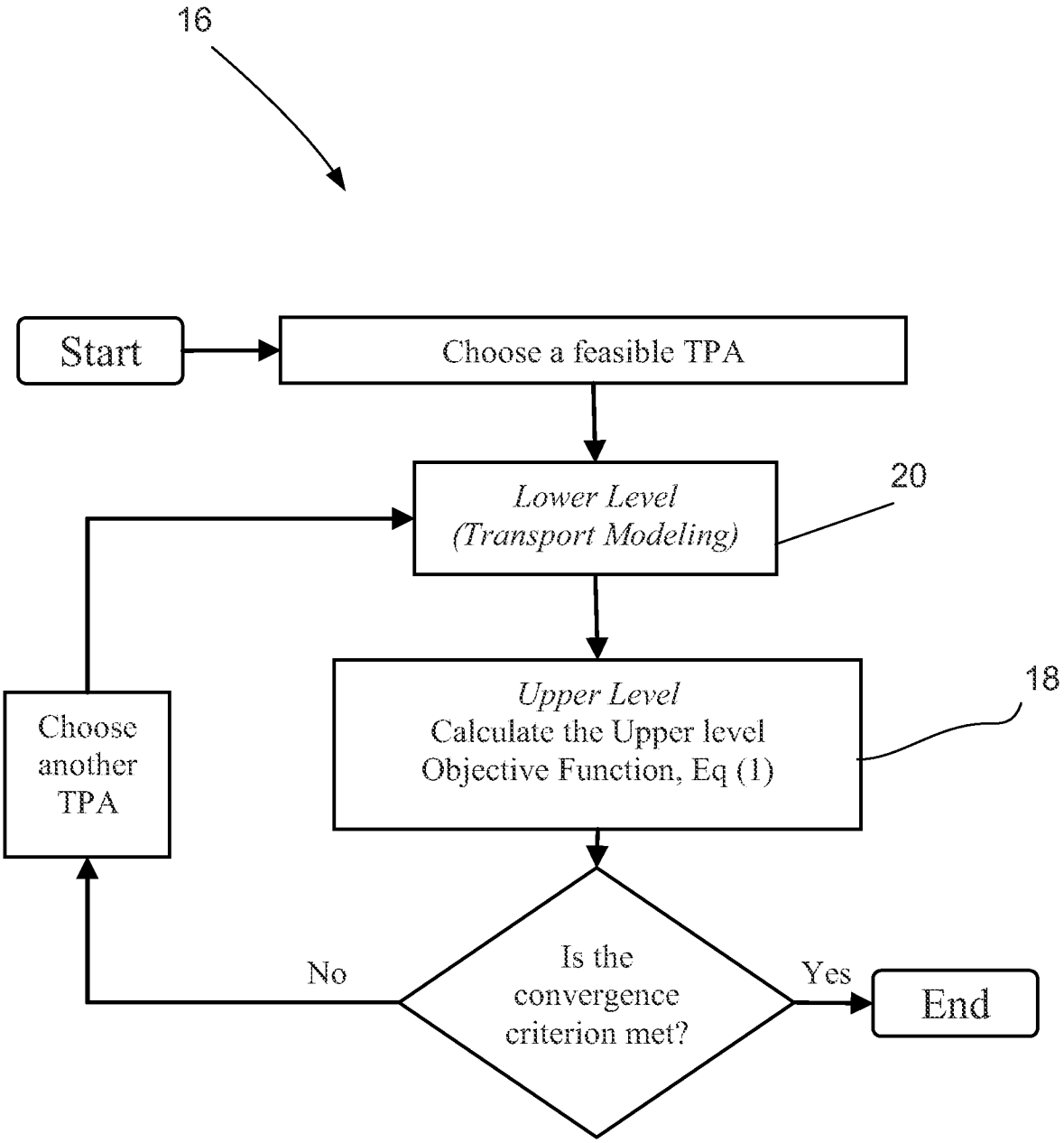
16
Start
Choose a feasible TPA
Lower Level
(Transport Modeling)
20
Upper Level
Calculate the Upper level
Objective Function, Eq (1)
18
Choose
another
TPA
Is the
convergence
criterion met?
No
Yes
End

Figure 2

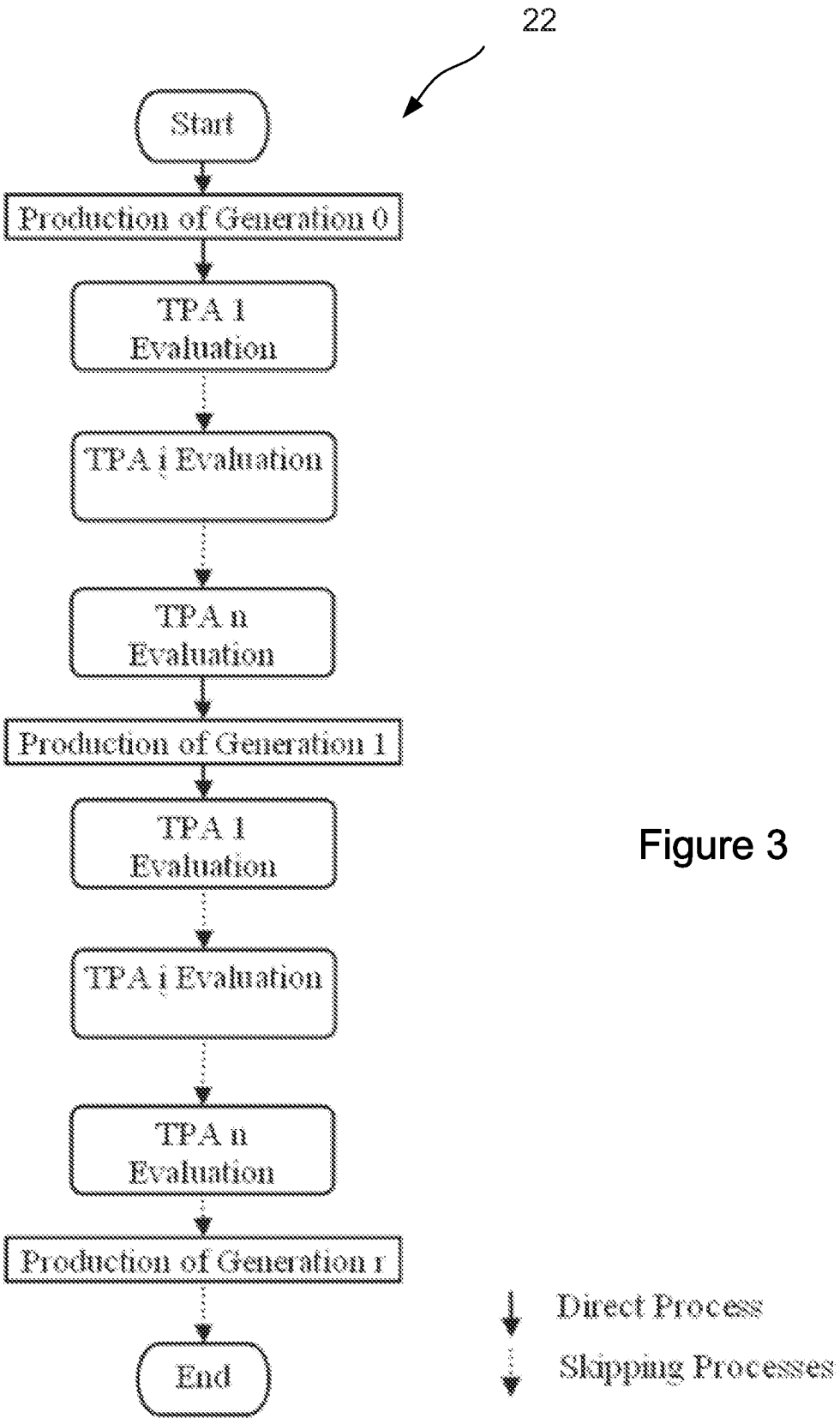
22
Start
Production of Generation 0
TPA 1 Evaluation
TPA i Evaluation
TPA n Evaluation
Production of Generation 1
TPA 1 Evaluation
TPA i Evaluation
TPA n Evaluation
Production of Generation r
End
Direct Process
Skipping Processes

Figure 3

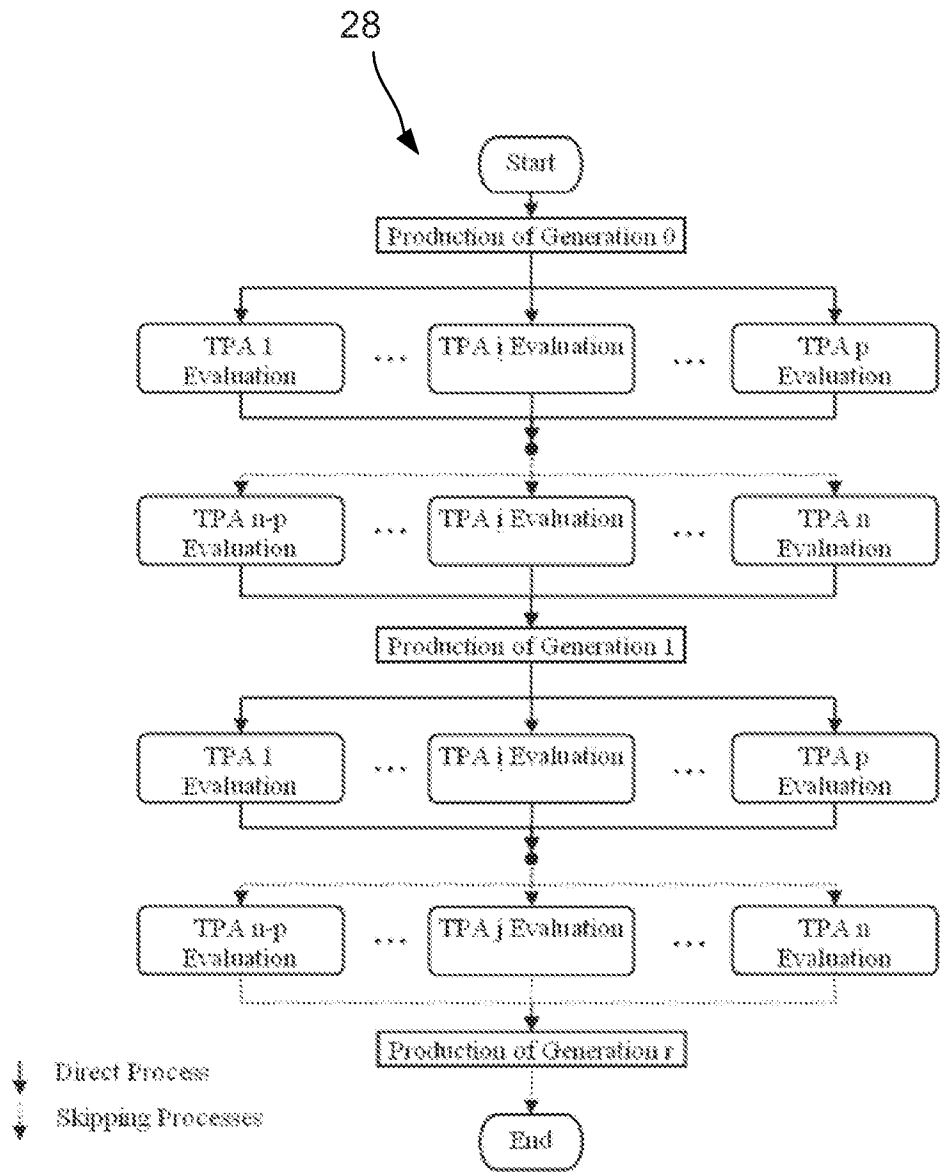
28
Start
Production of Generation 0
TPA 1 Evaluation
TPA j Evaluation
TPA p Evaluation
TPA n-p Evaluation
TPA j Evaluation
TPA n Evaluation
Production of Generation 1
TPA 1 Evaluation
TPA j Evaluation
TPA p Evaluation
TPA n-p Evaluation
TPA j Evaluation
TPA n Evaluation
Production of Generation r
End
Direct Process
Skipping Processes

Figure 4

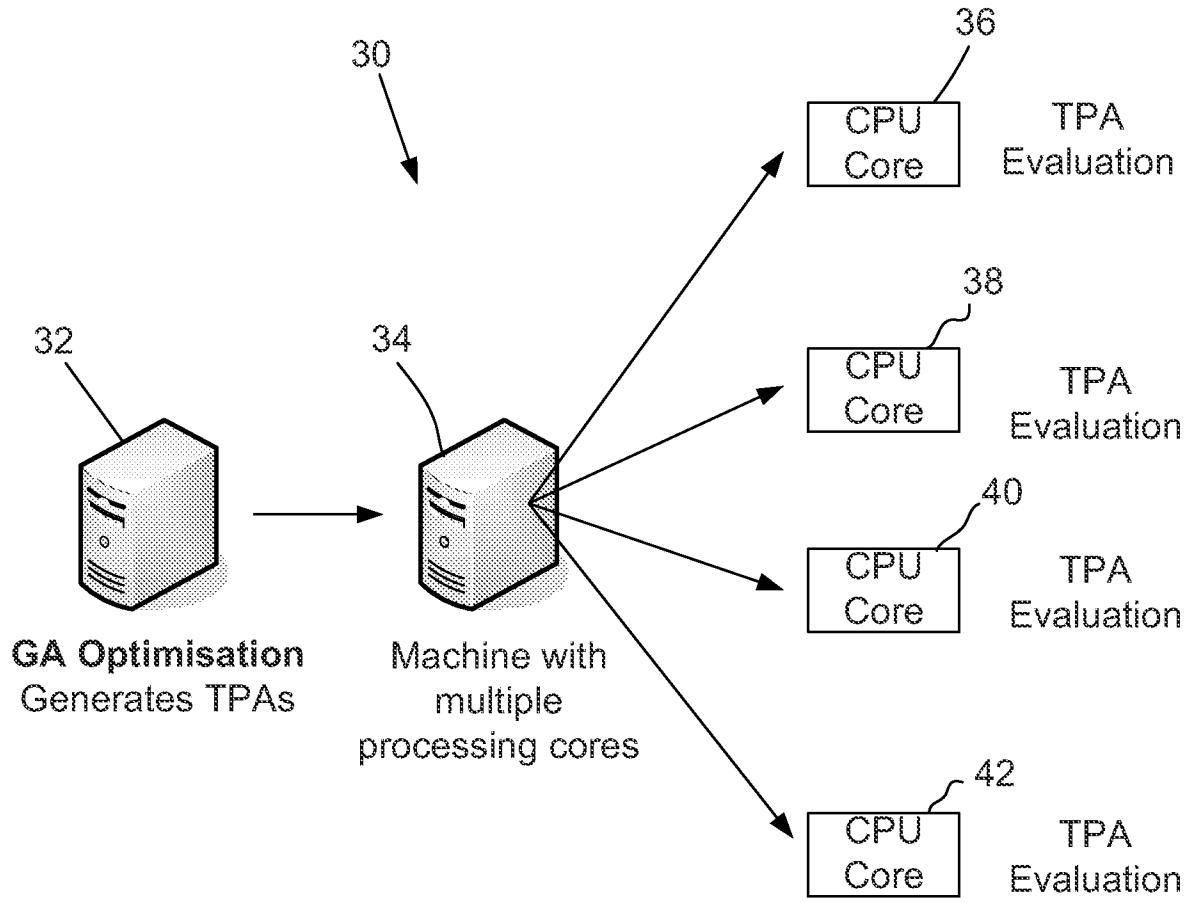
30
32
34
36
38
40
42
CPU Core
CPU Core
CPU Core
CPU Core
TPA Evaluation
TPA Evaluation
TPA Evaluation
TPA Evaluation
GA Optimisation
Generates TPAs
Machine with multiple processing cores

Figure 5

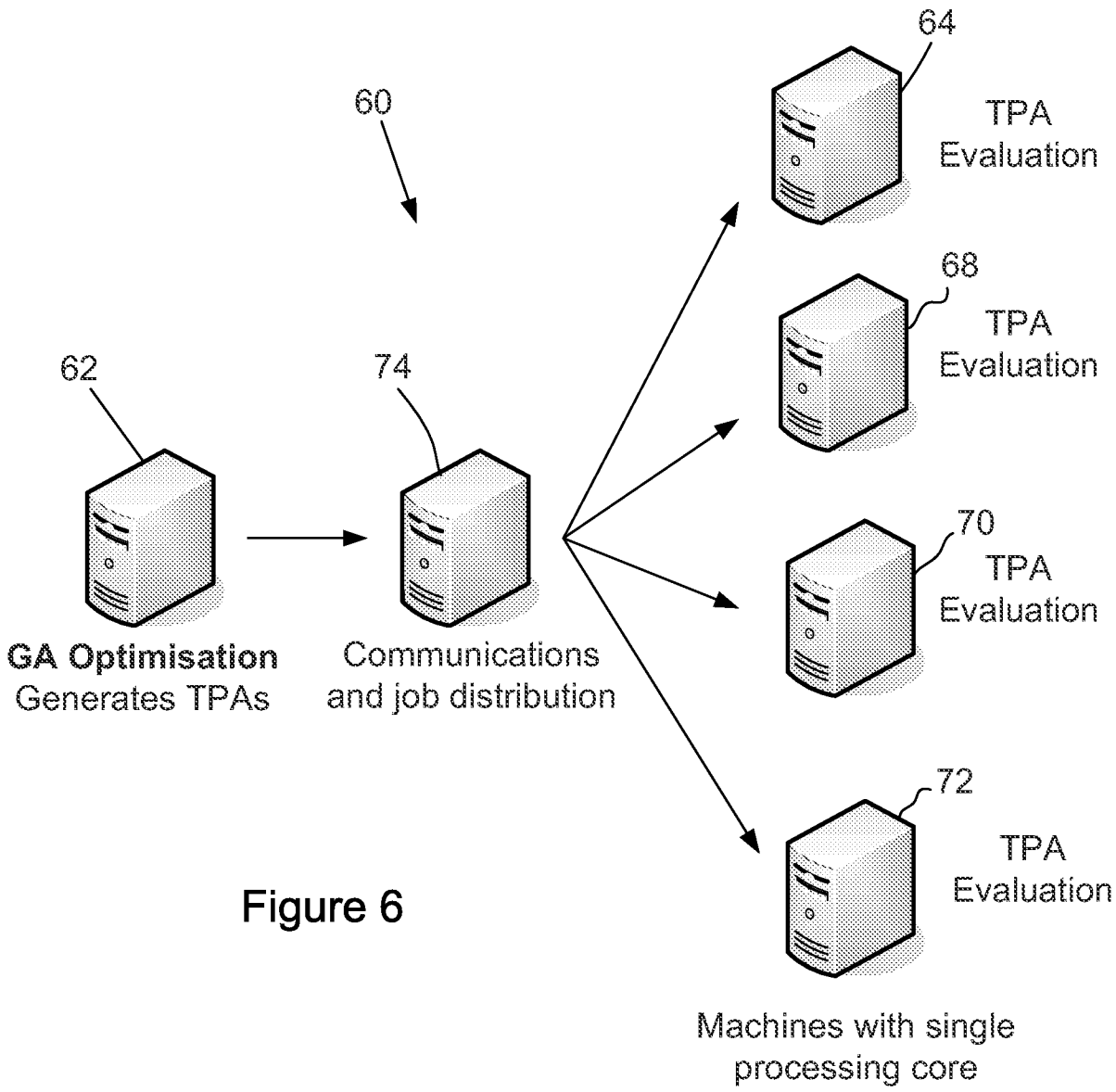
60
62
74
64
68
70
72
TPA Evaluation
TPA Evaluation
TPA Evaluation
TPA Evaluation
GA Optimisation
Generates TPAs
Communications and job distribution
Machines with single processing core

Figure 6

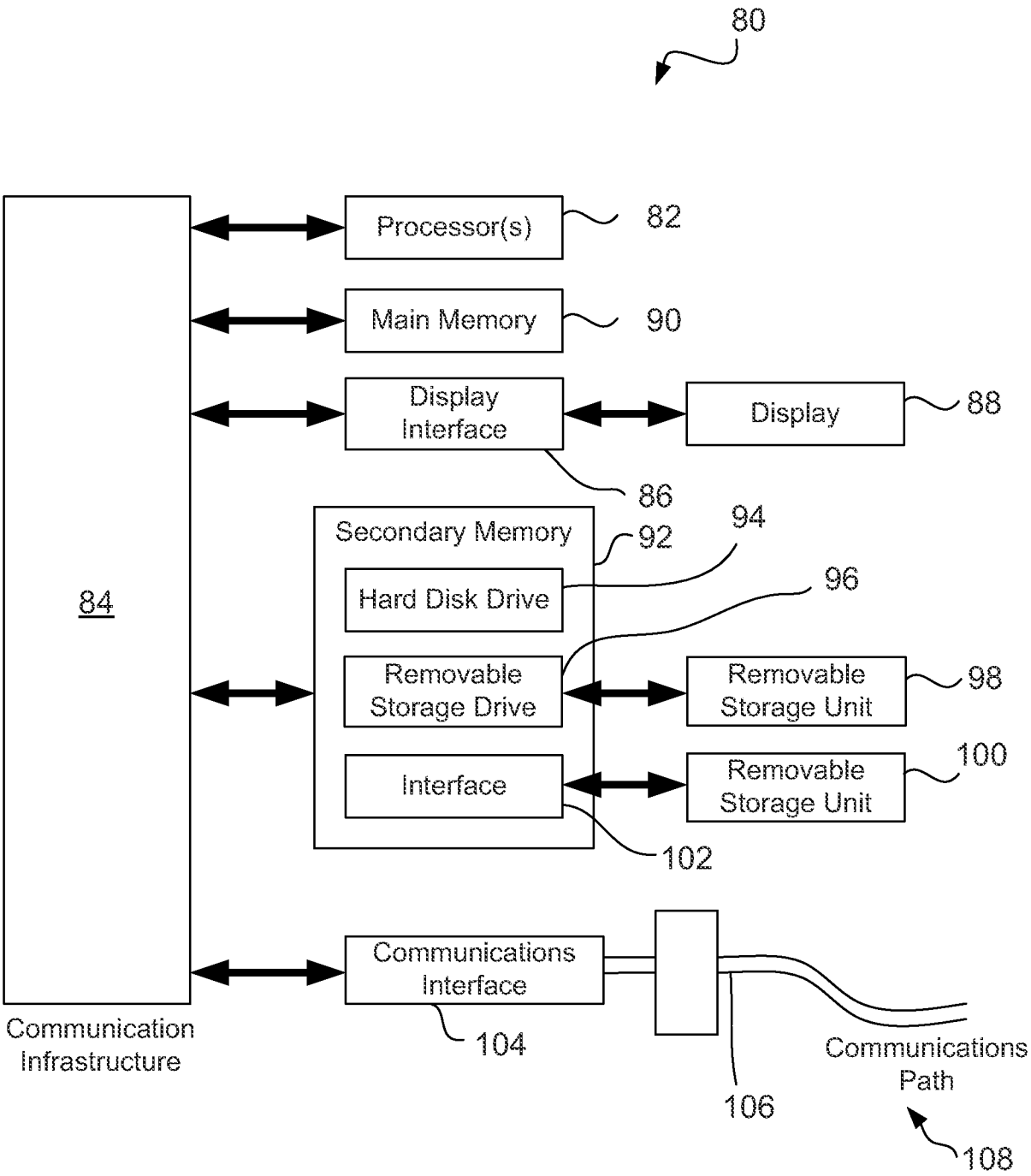
80
Processor(s)
82
Main Memory
90
Display Interface
Display
88
86
Secondary Memory
92
94
84
Hard Disk Drive
96
Removable Storage Drive
Removable Storage Unit
98
Interface
Removable Storage Unit
100
102
Communications Interface
104
106
Communications Path
108
Communication Infrastructure

Figure 7

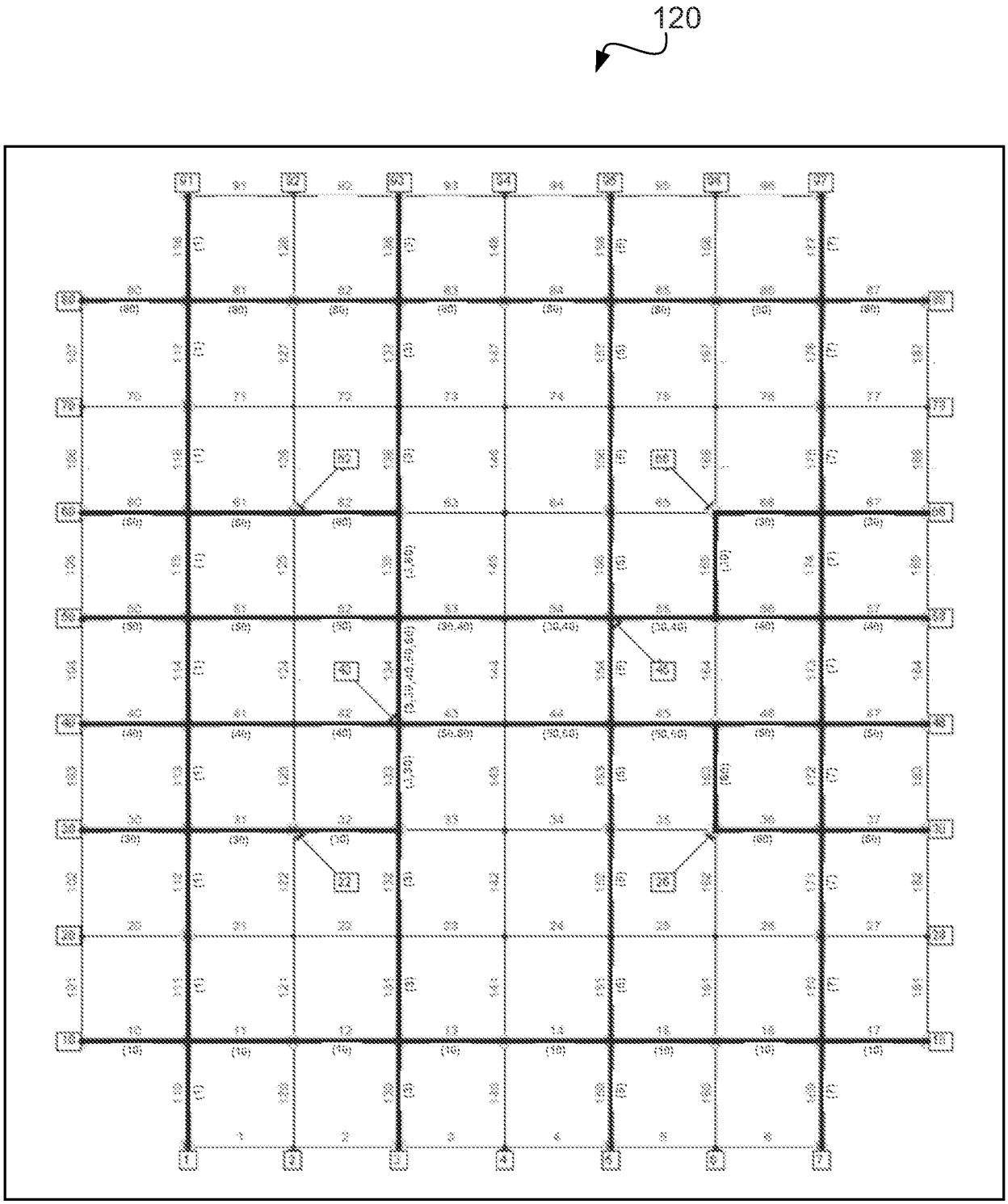

Figure 8

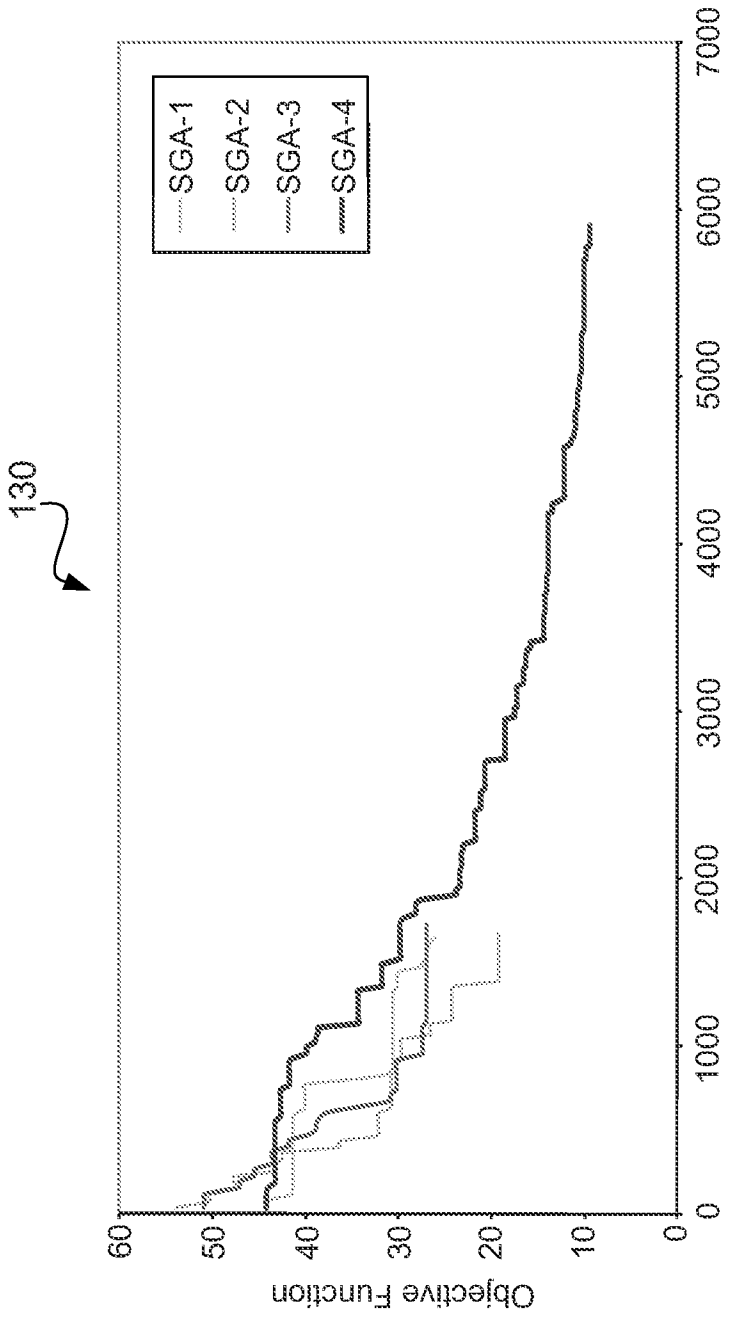
130
SGA-1
SGA-2
SGA-3
SGA-4
Objective Function
0
10
20
30
40
50
60
Number of Evaluations
0
1000
2000
3000
4000
5000
6000
7000

Figure 9

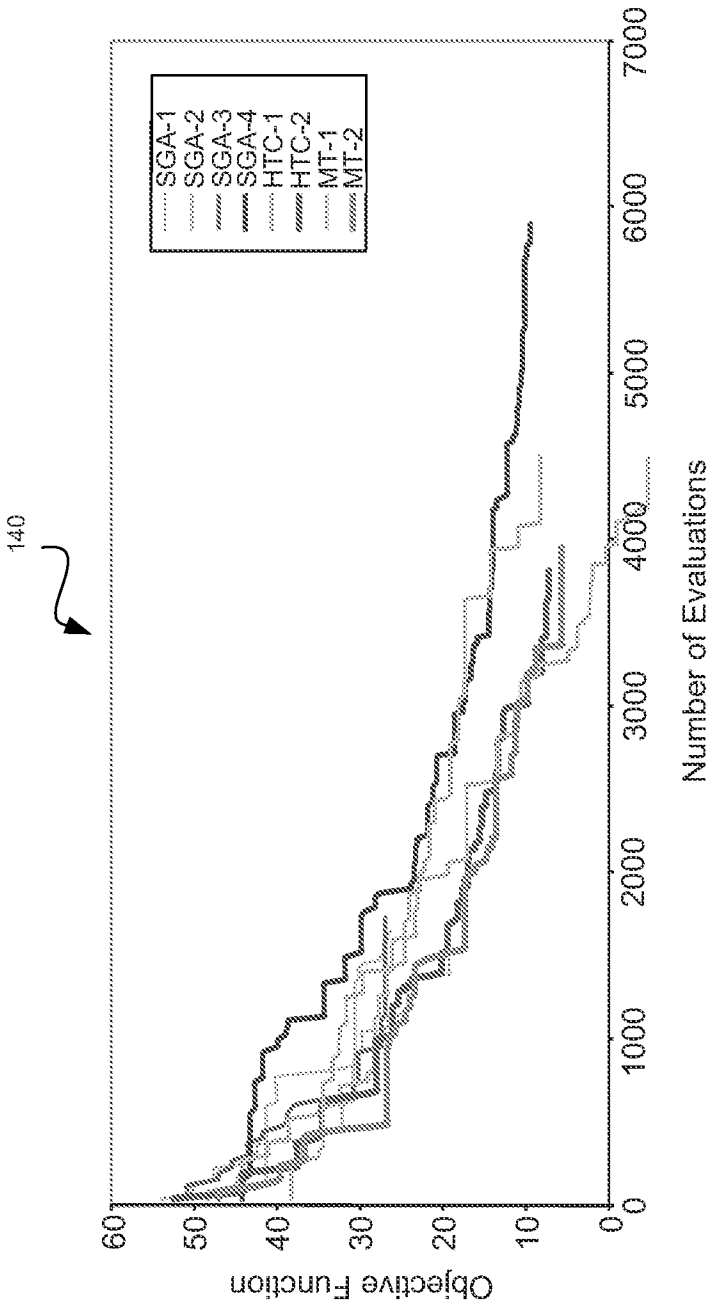
140
SGA-1
SGA-2
SGA-3
SGA-4
HTC-1
HTC-2
MT-1
MT-2
Objective Function
0
10
20
30
40
50
60
0
1000
2000
3000
4000
5000
6000
7000
Number of Evaluations

Figure 10

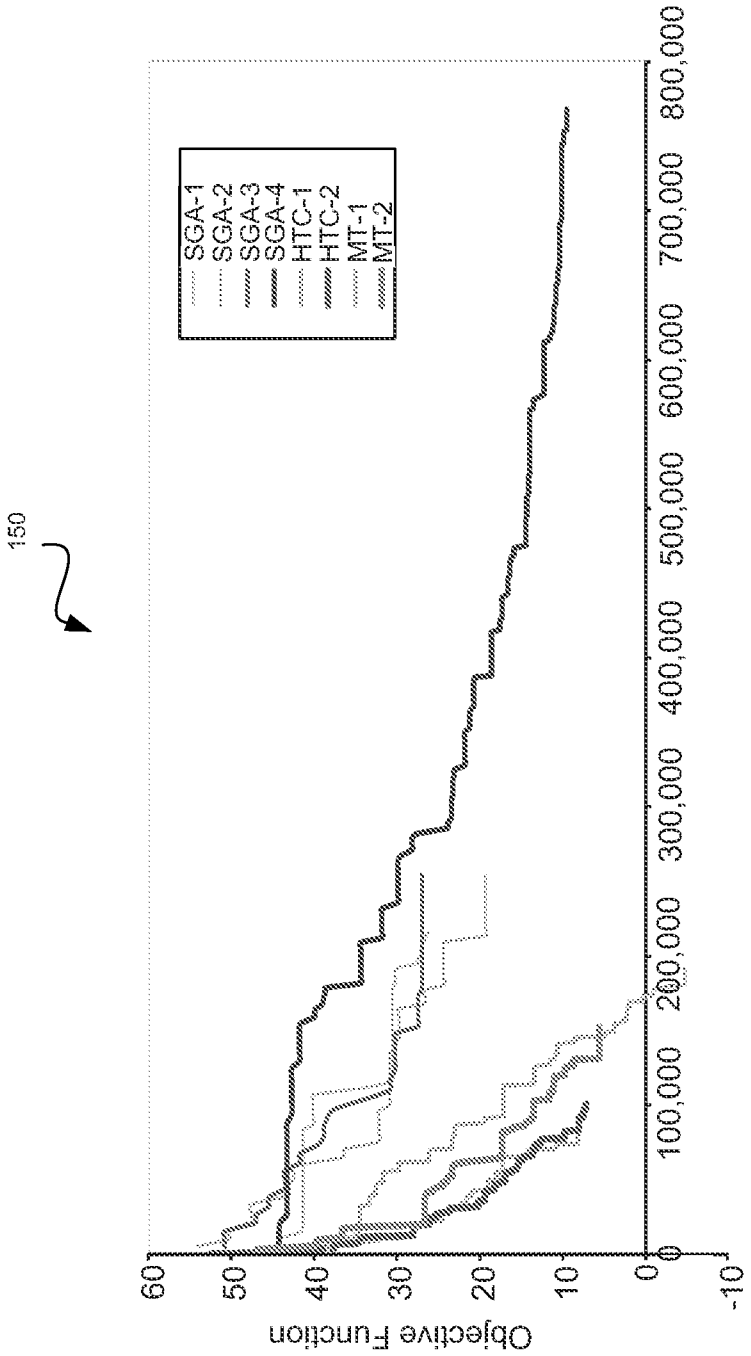
150
SGA-1
SGA-2
SGA-3
SGA-4
HTC-1
HTC-2
MT-1
MT-2
Objective Function
60
50
40
30
20
10
0
-10
0
100,000
200,000
300,000
400,000
500,000
600,000
700,000
800,000
Number of Evaluations

Figure 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU2012/000576

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/50 (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPAT, EPODOC, GOOGLE (keywords: transport, transit, optimise, prioritise, path and similar words)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	

[X] Further documents are listed in the continuation of Box C [ ] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 August 2012	16 August 2012

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustralia.gov.au
Facsimile No.: +61 2 6283 7999

Authorized officer

Tim Yang
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. 0262837923

INTERNATIONAL SEARCH REPORT		International application No.
C (Continuation).	DOCUMENTS CONSIDERED TO BE RELEVANT	**PCT/AU2012/000576**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Mesbah et al.; Policy-Making Tool for Optimization of Transit Priority Lanes in Urban Network; Transportation Research Record: Journal of the Transportation Research Board; Volume 2197; Issue 1; pages 54-62; 1 December 2010 see the entire document	1-11
X	Mesbah et al.; Optimization of transit priority in the transportation network using a decomposition methodology; 5 July 2010; retrieved from http://www.sciencedirect.com/science/article/pii/S0968090X1000104X on 13 August 2012 see the entire document	1-11