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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 January 2013 (03.01.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/003423 A1

(51) International Patent Classification:
G06F 17/11 (2006.01)

(21) International Application Number:
PCT/US2012/044334

(22) International Filing Date:
27 June 2012 (27.06.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/501,571 27 June 2011 (27.06.2011) US

(71) Applicant *(for all designated States except US)*: **KING ABDULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY** [SA/SA]; P.O. Box 55455, Jeddah, 21534 (SA).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **ROCKWOOD, Alyn** [US/SA]; King Abdullah University of Science and Technology, P.O. Box 55455, Jeddah, 21534 (SA).

(74) Agent: **GORDON, S., Scott**; Fulbright & Jaworski L.L.P., 98 San Jacinto Boulevard, Suite 1100, Austin, TX 78701 (US).

(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.

(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,

[Continued on next page]

(54) Title: SYSTEMS AND METHODS FOR INTERPOLATION-BASED DYNAMIC PROGRAMMING

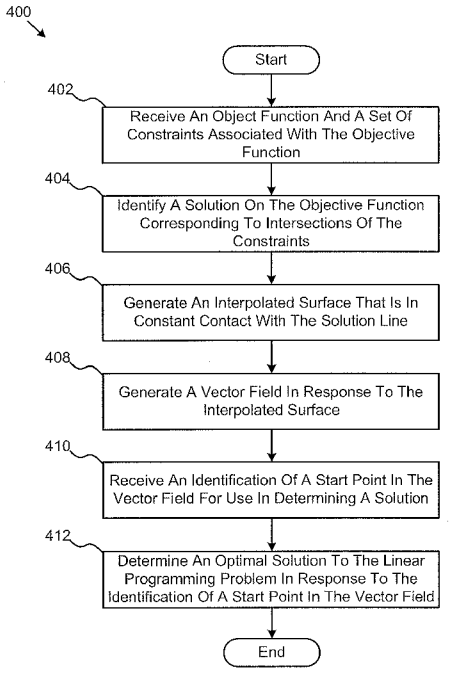

FIG. 4

(57) Abstract: Embodiments of systems and methods for interpolation-based dynamic programming. In one embodiment, the method includes receiving an object function and a set of constraints associated with the objective function. The method may also include identifying a solution on the objective function corresponding to intersections of the constraints. Additionally, the method may include generating an interpolated surface that is in constant contact with the solution. The method may also include generating a vector field in response to the interpolated surface.

WO 2013/003423 A1

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))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

DESCRIPTION

SYSTEMS AND METHODS FOR INTERPOLATION-BASED DYNAMIC PROGRAMMING

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0001] This invention relates to dynamic programming and more particularly relates to an apparatus system and method for interpolation-based dynamic programming.

DESCRIPTION OF THE RELATED ART

[0002] Dynamic Programming is a technique to find the optimal value of a objective function subject to constraints. FIG. 1 illustrates a common linear programming problem. In FIG. 1, the grayscale plane represents the objective function 102, and the black planes represent the constraints 104a-c used to define the problem. Linear programming is commonly used in business and industry such as in oil to identify the most profitable refinery-wide operating strategy. Classical methods like Simplex and Interior-Point have larger than cubic time complexity in worst cases.

[0003] One of ordinary skill in the art will recognize various embodiments of processes which may be classified as "Simplex" methods. Simplex methods are generally quite accurate relative to other dynamic programming techniques, but are typically too slow and resource intensive for higher dimension problems. Similarly, one of ordinary skill in the art will recognize a variety of embodiments which may be classified as "Interior-Point" methods. These methods represent the opposite end of the spectrum relative to Simplex methods, because they are generally quite fast and resource-light, but they are relatively inaccurate as compared to, for example, Simplex methods.

SUMMARY OF THE INVENTION

[0004] Embodiments of methods for interpolation-based dynamic programming are described. In one embodiment, the method includes receiving an objective function and a set of constraints associated with the objective function. The method may also include identifying a solution on the objective function corresponding to intersections of the constraints. Additionally, the method may

include generating an interpolated surface that is in constant contact with the solution. The method may also include generating a vector field in response to the interpolated surface.

[0005] In one embodiment, the method may also include receiving an identification of a start point in the vector field. Additionally, the method may include determining an optimal solution to the dynamic programming problem in response to the identification of a start point in the vector field. Determining the optimal solution comprises following the vectors in the vector field from a start position along a contour of the solution until an optimal solution is reached.

[0006] In one embodiment, determining the optimal solution may include following a first line from the start point to a first intersection with the solution, the direction of the line determined in response to the direction of the vectors in the vector field, and following a second line to a second intersection with the solution, the direction of the second line being parallel to the portion of the solution in which the first intersection was reached and determined in response to the direction of the vectors in the vector field. Such an embodiment may include following a plurality of additional lines to a plurality of additional intersections with the solution until an optimal solution is reached, the direction of the plurality of additional lines being parallel to the portion of the solution in which a preceding intersection was reached and determined in response to the direction of the vectors in the vector field. In one embodiment, the second line is separated from the portion of the solution in which the first intersection was reached by a backoff value. The backoff value will be in the interval (0,1), which may be considered the fraction of the distance between current point and intersection point.

[0007] In a further embodiment, the method may include steps for determining an intermediate solution to the dynamic programming problem in response to the identification of a start point in the vector field, and determining an optimum solution according to a Simplex algorithm, wherein the Simplex algorithm uses the intermediate solution as an input to determine the optimum solution. In another embodiment, the method the start point received is an intermediate solution determined by use of an Interior-Point algorithm.

[0008] Embodiments of systems for interpolation-based dynamic programming are also described. In one embodiment, the system includes an input interface configured to receive an object function and a set of constraints associated with the objective function. The system may also include a memory device coupled to the input interface and configured to store the object

function and the set of constraints. Also, the system may include a processor coupled to the memory device. The processor may be configured to identify a solution on the objective function corresponding to intersections of the constraints, generate an interpolated surface that is in constant contact with the solution, and generate a vector field in response to the interpolated surface.

[0009] Embodiments of computer program products comprising a computer readable medium having computer usable program code executable to perform operations for interpolation-based linear programming are also described. In one embodiment the operations include receiving an object function and a set of constraints associated with the objective function. The operations may also include identifying a solution on the objective function corresponding to intersections of the constraints. Additionally, the operations may include generating an interpolated surface that is in constant contact with the solutions. The operations may also include generating a vector field in response to the interpolated surface.

[0010] The term "coupled" is defined as connected, although not necessarily directly, and not necessarily mechanically.

[0011] The terms "a" and "an" are defined as one or more unless this disclosure explicitly requires otherwise.

[0012] The term "substantially" and its variations are defined as being largely but not necessarily wholly what is specified as understood by one of ordinary skill in the art, and in one non-limiting embodiment "substantially" refers to ranges within 10%, preferably within 5%, more preferably within 1%, and most preferably within 0.5% of what is specified.

[0013] The terms "comprise" (and any form of comprise, such as "comprises" and "comprising"), "have" (and any form of have, such as "has" and "having"), "include" (and any form of include, such as "includes" and "including") and "contain" (and any form of contain, such as "contains" and "containing") are open-ended linking verbs. As a result, a method or device that "comprises," "has," "includes" or "contains" one or more steps or elements possesses those one or more steps or elements, but is not limited to possessing only those one or more elements. Likewise, a step of a method or an element of a device that "comprises," "has," "includes" or "contains" one or more features possesses those one or more features, but is not limited to possessing only those one or more features. Furthermore, a device or structure that is configured

in a certain way is configured in at least that way, but may also be configured in ways that are not listed.

[0014] Other features and associated advantages will become apparent with reference to the following detailed description of specific embodiments in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The following drawings form part of the present specification and are included to further demonstrate certain aspects of the present invention. The invention may be better understood by reference to one or more of these drawings in combination with the detailed description of specific embodiments presented herein.

[0016] FIG. 1 illustrates an objective function (the grayscale plane) and constraints (black planes) according to a common linear programming example.

[0017] FIG. 2 illustrates one embodiment of a method for computing a dynamic programming solution using an interpolant that passes through intersections of an objective surface and constraints.

[0018] FIG. 3 is a schematic block diagram illustrating one embodiment of a computer system configurable to perform one embedment of a method for interpolation-based linear programming.

[0019] FIG. 4 is a schematic flowchart diagram illustrating one embodiment of a method for interpolation-based dynamic programming.

[0020] FIG. 5 illustrates one embodiment of a gradient field used according to one embodiment of a method for determining a dynamic programming solution.

[0021] FIG. 6 illustrates an example of a Steepest Descent method to find a dynamic programming solution used as a comparison for validation of the present embodiments.

[0022] FIG. 7A illustrates a sample solution route with a first start point.

[0023] FIG. 7B illustrates a sample solution route with a second start point.

[0024] FIG. 8 illustrates how different start points may generate different solution routes.

[0025] FIG. 9 illustrates time cost of an embodiment of the method under different numbers of constraints.

[0026] FIG. 10A - 10B is a schematic flow chart diagram illustrating one embodiment of a method for interpolation-based dynamic programming.

DETAILED DESCRIPTION

[0027] Various features and advantageous details are explained more fully with reference to the nonlimiting embodiments that are illustrated in the accompanying drawings and detailed in the following description. Descriptions of well known starting materials, processing techniques, components, and equipment are omitted so as not to unnecessarily obscure the invention in detail. It should be understood, however, that the detailed description and the specific examples, while indicating embodiments of the invention, are given by way of illustration only, and not by way of limitation. Various substitutions, modifications, additions, and/or rearrangements within the spirit and/or scope of the underlying inventive concept will become apparent to those skilled in the art from this disclosure.

[0028] Certain units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. A module is "[a] self-contained hardware or software component that interacts with a larger system. Alan Freedman, "The Computer Glossary" 268 (8th ed. 1998). A module comprises a machine or machines executable instructions. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

[0029] Modules may also include software-defined units or instructions, that when executed by a processing machine or device, transform data stored on a data storage device from a first state to a second state. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations

which, when joined logically together, comprise the module, and when executed by the processor, achieve the stated data transformation.

[0030] Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices.

[0031] In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of the present embodiments. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

[0032] The present embodiments are directed to methods and systems for interpolation-based dynamic programming. In one embodiment, an interpolation surface 202 is introduced to pass through the intersections of an objective surface 102 and constraints 104a-c. The interpolant surface 202 may be in constant contact with a solution 204 defined by the intersections of the objective surface 102 and constraints 104a-c. FIG. 2 illustrates one embodiment of a method for computing a dynamic programming solution using an interpolant surface 202 that passes through intersections of an objective surface 102 and constraints 104a-c. The subsequent gradient field underpins several possible algorithms. For example, the gradient may be used as the direction of a projection to an image point on a constraint plane. The image point may then be retracted on the route with a backoff value and this may become a new start point. It iterates from that point until convergence at an optimum solution. An example of this process is described in greater detail with reference to FIG. 5 below. Results and analysis show that the interpolation method may stably attain quadratic time complexity.

[0033] FIG. 3 illustrates a computer system 300 adapted to perform one or more embodiments of a method for interpolation-based dynamic programming. In such an embodiment, the central processing unit (CPU) 302 is coupled to the system bus 304. The CPU 302 may be a general purpose CPU or microprocessor. The present embodiments are not restricted by the architecture of the CPU 302, so long as the CPU 302 supports the modules and operations as described herein. The CPU 302 may execute various logical instructions according to the present embodiments. For example, the CPU 302 may execute machine-level instructions according to the exemplary operations described below with reference to FIGS. 4 and 10.

[0034] The computer system 300 also may include Random Access Memory (RAM) 308, which may be SRAM, DRAM, SDRAM, or the like. The computer system 300 may utilize RAM 308 to store the various data structures used by a software application configured to perform steps of a method for interpolation-based dynamic programming according to the present embodiments. The computer system 300 may also include Read Only Memory (ROM) 306 which may be PROM, EPROM, EEPROM, optical storage, or the like. The ROM may store configuration information for booting the computer system 300. The RAM 308 and the ROM 306 hold user and system 100 data.

[0035] The computer system 300 may also include an input/output (I/O) adapter 310, a communications adapter 314, a user interface adapter 316, and a display adapter 322. The I/O adapter 310 and/or user the interface adapter 316 may, in certain embodiments, enable a user to interact with the computer system 300 in order to input information, including selection of objective functions, constraints, and start points. In a further embodiment, the display adapter 322 may display a graphical user interface associated with a software or web-based application for interpolation-based dynamic programming.

[0036] The I/O adapter 310 may connect to one or more storage devices 312, such as one or more of a hard drive, a Compact Disk (CD) drive, a floppy disk drive, a tape drive, to the computer system 300. The communications adapter 314 may be adapted to couple the computer system 300 to the network 106, which may be one or more of a LAN and/or WAN, and/or the Internet. The user interface adapter 316 couples user input devices, such as a keyboard 320 and a pointing device 318, to the computer system 300. The display adapter 322 may be driven by the CPU 302 to control the display on the display device 324.

[0037] The present embodiments are not limited to the architecture of system 300. Rather the computer system 300 is provided as an example of one type of computing device that may be adapted to perform the functions of system for interpolation-based dynamic programming. For example, any suitable processor-based device may be utilized including without limitation, including personal data assistants (PDAs), computer game consoles, and multi-processor servers. Moreover, the present embodiments may be implemented on application specific integrated circuits (ASIC) or very large scale integrated (VLSI) circuits. In fact, persons of ordinary skill in the art may utilize any number of suitable structures capable of executing logical operations according to the described embodiments.

[0038] FIG. 4 is a schematic flowchart diagram illustrating one embodiment of a method 400 for interpolation-based dynamic programming. In one embodiment, the method 400 starts when the input adapter 310 receives 402 an object function 102 and a set of constraints 104a-c associated with the objective function. The CPU 302 may then identify 404 a solution 204 on the objective function 102 corresponding to intersections of the constraints 104a-c. Additionally, the CPU 302 may generate 406 an interpolated surface 202 that is in constant contact with the solution 204. In a further embodiment, the CPU 302 may generate 408 a vector field 502 in response to the interpolated surface 202.

[0039] In one embodiment, the input adapter 310 may also receive 410 an identification of a start point 504 in the vector field 502 for use in determining a solution. In an alternative embodiment, the start point 504 may be selected at random. In still another embodiment, a present start point 504 may be selected. In yet another embodiment, the start point 504 may be automatically selected in response to one or more preset rules or conditions.

[0040] Additionally, the CPU 302 may determine 412 an optimal solution to the linear programming problem in response to the identification of a start point 504 in the vector field. For example, determining the optimal solution comprises following the vectors in the vector field 502 from a start position 504 along a contour of the solution 204 until an optimal solution is reached.

[0041] FIG. 5 illustrates one embodiment of a vector field 502 used according to one embodiment of a method for deterring a dynamic programming solution. In one embodiment, the vector field 502 may be generated by the CPU 302. In such an embodiment, the CPU 302 may compute a gradient of the interpolated surface 202. The gradient of the interpolated surface 202

may yield a plurality of vectors, each vector pointing in a direction corresponding to the slope of the interpolated surface 202. The vector field may additionally include the solution. In one embodiment, the vectors in the vector field will continuously point to a portion of the solution 204.

[0042] FIG. 6 illustrates an example of a Steepest Descent method to find a dynamic programming solution used as a comparison for validation of the present embodiments. Steepest Descent is another method to use the direction of every search, but takes much more time than the previous approach. Compared to steps of steepest descent method, the method described in FIG. 5 is much faster within an acceptable accuracy.

[0043] FIG. 7A illustrates a sample solution route with a first start point 504. In the depicted embodiment, the start point 504 may be [0.1 0.1 0.5]. In one embodiment, determining the optimal solution may include following a first line 702 from the start point to a first intersection with the solution 204, the direction of the line 702 determined in response to the direction of the vectors in the vector field 502, and following a second line 704 to a second intersection with the solution 204, the direction of the second line being nearly parallel to the portion of the solution 204 in which the first intersection was reached and determined in response to the direction of the vectors in the vector field. Such an embodiment may include following a plurality of additional lines to a plurality of additional intersections with the solution 204 until an optimal solution is reached, the direction of the plurality of additional lines being nearly parallel to the portion of the solution 204 in which a preceding intersection was reached and determined in response to the direction of the vectors in the vector field 502.

[0044] In one embodiment, the second line is separated from the portion of the solution in which the first intersection was reached by a back up value. In the embodiment illustrated in FIG. 7A, after four iterations the algorithm converges at an optimal solution with accuracy of 2^{-10}. In such an embodiment, the backoff value may be 0.8.

[0045] FIG. 7B illustrates another sample solution route starting from a second start point 504. In this example, the start point 504 is [0.3 0.1 0.4]. As illustrated, after 5 iterations the algorithm converges at an optimal solution with accuracy 2^{-10}. In this embodiment, the backoff value is also 0.8; however, one of ordinary skill in the art will recognize alternative back up values that may be suitable for use with the present embodiments.

[0046] FIG. 8 illustrates how different start points 504 may generate different solution routes. In each embodiment, it can be seen that the first step is to follow a straight path from the start point 504 to an intersection with the solution 204. The direction of the straight line may be calculated in response to the directions of the vectors along the path. For example, an average of the vector directions may be calculated. One of ordinary skill in the art may recognize alternative regression techniques for determining the direction of the path in response to the vectors in the vector field 502. In one embodiment, subsequent paths may overshoot the optimal solution, in such an embodiment; however in each case, the vectors in the vector field lead 502 to the optimal solution in subsequent steps. Depending upon the starting point 504 and the complexity of the problem, the solution may be identified in as little as one iteration, or many more iterations. FIG. 9 illustrates time cost of an embodiment of the method under different numbers of constraints. Tests of the described embodiments conducted on 3D benchmarks showed a time complexity of O(n2L) for general and specific cases, compared with O(n3.5L) in worst cases using Simplex or Interior Point method, 'L' represents input size, such as the dimension or bits precision, and 'n' represents the number of constraints. Additionally, test results showed fast convergence on the optimal solution. In most cases, only 5 or 6 iterations, at most, were required to attain the optimal value with accuracy 2−10.

[0047] In one embodiment, embodiments of the presently described process may be used in conjunction with Simplex methods and/or Interior-Point methods. For example, an intermediate solution may be found quickly using Interior-Point methods, and then the intermediate solution may be used as a starting point 504 for the processes described in FIGs. 4-7. Alternatively, the processes described FIGs. 4-7 may be used to determine an intermediate point, and that intermediate point may be used as a starting point for a Simplex method, which tends to be very accurate, but too time intensive for higher-dimension problems. In still a further embodiment, all three methods may be combined such that an Interior-point method is used to determine a first intermediate solution, the processes of FIGs. 4-7 are used to determine a second intermediate solution, and a Simplex method is used to determine an optimum solution. One of ordinary skill in the art will recognize various embodiments of Interior-Point methods and Simplex methods that may be used in conjunction with the present embodiments.

[0048] For example, the method 400 of FIG. 4 may be expanded to include steps for determining an intermediate solution to the dynamic programming problem in response to the identification of a start point in the vector field, and determining an optimum solution according

to a Simplex algorithm, wherein the Simplex algorithm uses the intermediate solution as an input to determine the optimum solution. In another embodiment, the method 400 the start point received at block 410 is an intermediate solution determined by use of an Interior-Point algorithm.

[0049] FIG. 10 is a flowchart diagram illustratingpseudo code for implementing one embodiment of a method for interpolation-based dynamic programming. In one embodiment, the pseudo code for this method can be implemented in various computer programming languages, including Fortran, Pascal, C, C++, C#, Perl, Java, or the like. In a further embodiment, the pseudo code may be implemented in a math programming language such as Matlab, Mathematica, or the like. In such embodiments, the pseudocode may be stored in RAM 308 and/or ROM 306. The pseudo code may be loaded into a processing device, such as CPU 302, and executed. When executed by a processing device, the code may cause the processing device to perform operations as described in the flowchart of FIG. 10.

[0050] For example, in one embodiment, the method starts when the input adapter 310 and/or CPU 302 receives an input at block 1002. In one embodiment, the input is a d-dimensional cost function $\mathscr{F}$ and n constraints associated with the cost function. In one embodiment, an interpolation function $\mathscr{I}$is defined at block 1004, which passes through the n intersections of the Constraints and the Cost Function $\mathscr{F}$as shown at block 1008. The method may also be terminated at block 1010 as terminating conditions are understood by those skilled in the art. If the terminating condition is met at block 1010, then an output value may be generated at block 1012. In one embodiment, the CPU 302 computes the gradient using partial derivatives of interpolation surface $\mathscr{I}$, or finite difference approximation to derivatives as shown in block 1014.

[0051] As shown in FIG. 10B, if the maximum number of constraints have not been used in identification of the optimum solution, as determined at block 1016, the CPU 302 may compute the new step point as shown in blocks 1018-1120 by finding a point which is a percentage of the distance between the current point **pnt** and the intersection point **isect** as shown at blocks 1022-1032. In one embodiment, the specified method continues to find a new step point until any terminating condition is met at block 1010. In one embodiment, an optimal value is found at block 1012 and displayed by Display Adapter 322. In one embodiment, the optimal value found can achieve the smallest cost for a minimum problem or the greatest profit for a maximum

problem. In one embodiment, the solved optimal value for a minimum problem is distinguished from a maximum problem by a minus in all equations.

[0052] In various embodiments, the CPU 302 may communicate certain inputs and or outputs generated by the code as illustrated in FIG. 10 with an I/O adapter 310, a communications adapter 314, a user interface adapter 316 and/or a display adapter 322. One of ordinary skill in the art will recognize a variety of options for communicating inputs and outputs with users, data storage devices 312 and/or other system computers.

[0053] All of the methods disclosed and claimed herein can be made and executed without undue experimentation in light of the present disclosure. While the apparatus and methods of this invention have been described in terms of preferred embodiments, it will be apparent to those of skill in the art that variations may be applied to the methods and in the steps or in the sequence of steps of the method described herein without departing from the concept, spirit and scope of the invention. In addition, modifications may be made to the disclosed apparatus and components may be eliminated or substituted for the components described herein where the same or similar results would be achieved. All such similar substitutes and modifications apparent to those skilled in the art are deemed to be within the spirit, scope, and concept of the invention as defined by the appended claims.

CLAIMS

1. A method for interpolation-based dynamic programming, the method comprising:
receiving an objective function and a set of constraints associated with the objective
identifying a solution on the objective function corresponding to intersections of the constraints;
generating an interpolated surface that is in constant contact with the solution, and the objective function; and
generating a vector field in response to the interpolated surface.

2. The method of claim 1, further comprising receiving an identification of a start point in the vector field.

3. The method of claim 2, further comprising determining an optimal solution to the dynamic programming problem in response to the identification of a start point in the vector field.

4. The method of claim 3, wherein determining the optimal solution comprises following the vectors in the vector field from a start position along a contour of the solution until an optimal solution is reached.

5. The method of claim 3, wherein determining the optimal solution comprises:
following a first line from the start point to a first intersection with the solution, the direction of the line determined in response to the direction of the vectors in the vector field;
following a second line to a second intersection with the solution, the direction of the second line being substantially parallel to the portion of the solution in which the first intersection was reached and determined in response to the direction of the vectors in the vector field;
following a plurality of additional lines to a plurality of additional intersections with the solution until an optimal solution is reached, the direction of the plurality of additional lines being parallel to the portion of the solution in which a preceding intersection was reached and determined in response to the direction of the vectors in the vector field.

6. The method of claim 5, wherein the second line is separated from the portion of the solution in which the first intersection was reached by a back value.

6.1. The method of claim 2, further comprising:

determining an intermediate solution to the dynamic programming problem in response to the identification of a start point in the vector field; and determining an optimum solution according to a Simplex algorithm, wherein the Simplex algorithm uses the intermediate solution as an input to determine the optimum solution.

6.2. The method of claim 2, wherein the start point is an intermediate solution determined by use of an Interior-Point algorithm.

7. A system for interpolation-based dynamic programming, the system comprising:

an input interface configured to receiving an object function and a set of constraints associated with the objective function;

a memory device coupled to the input interface and configured to store the object function and the set of constraints; and

a processor coupled to the memory device, the processor configured to:

identify a solution on the objective function corresponding to intersections of the constraints;

generate an interpolated surface that is in constant contact with the solution; and

generate a vector field in response to the interpolated surface.

8. The system of claim 7, wherein the input interface is further configured to receive an identification of a start point in the vector field.

9. The system of claim 8, wherein the processor is further configured to determine an optimal solution to the dynamic programming problem in response to the identification of a start point in the vector field.

10. The system of claim 9, wherein the processor performs operations to follow the vectors in the vector field from a start position along a contour of the solution until an optimal solution is reached.

11. The system of claim 11, wherein the processor further performs operations comprising:

following a first line from the start point to a first intersection with the solution, the direction of the line determined in response to the direction of the vectors in the vector field;

following a second line to a second intersection with the solution, the direction of the second line being parallel to the portion of the solution in which the first intersection was reached and determined in response to the direction of the vectors in the vector field;

following a plurality of additional lines to a plurality of additional intersections with the solution until an optimal solution is reached, the direction of the plurality of additional lines being parallel to the portion of the solution in which a preceding intersection was reached and determined in response to the direction of the vectors in the vector field.

12. The computer program product of claim 11, wherein the second line is separated from the portion of the solution in which the first intersection was reached by a backoff value.

13. A computer program product comprising a computer readable medium having computer usable program code executable to perform operations for interpolation-based dynamic programming, the operations of the computer program product comprising:

receiving an object function and a set of constraints associated with the objective function;

identifying a solution on the objective function corresponding to intersections of the constraints;

generating an interpolated surface that is in constant contact with the solution; and

generating a vector field in response to the interpolated surface.

14. The computer program product of claim 13, further comprising receiving an identification of a start point in the vector field.

15. The computer program product of claim 14, further comprising determining an optimal solution to the dynamic programming problem in response to the identification of a start point in the vector field.

16. The computer program product of claim 15, wherein determining the optimal solution comprises following the vectors in the vector field from a start position along a contour of the solution until an optimal solution is reached.

17. The computer program product of claim 15, wherein determining the optimal solution comprises:

following a first line from the start point to a first intersection with the solution, the direction of the line determined in response to the direction of the vectors in the vector field;

following a second line to a second intersection with the solution, the direction of the second line being parallel to the portion of the solution in which the first intersection was reached and determined in response to the direction of the vectors in the vector field;

following a plurality of additional lines to a plurality of additional intersections with the solution until an optimal solution is reached, the direction of the plurality of additional lines being parallel to the portion of the solution in which a preceding intersection was reached and determined in response to the direction of the vectors in the vector field.

18. The computer program product of claim 17, wherein the second line is separated from the portion of the solution in which the first intersection was reached by a back value.

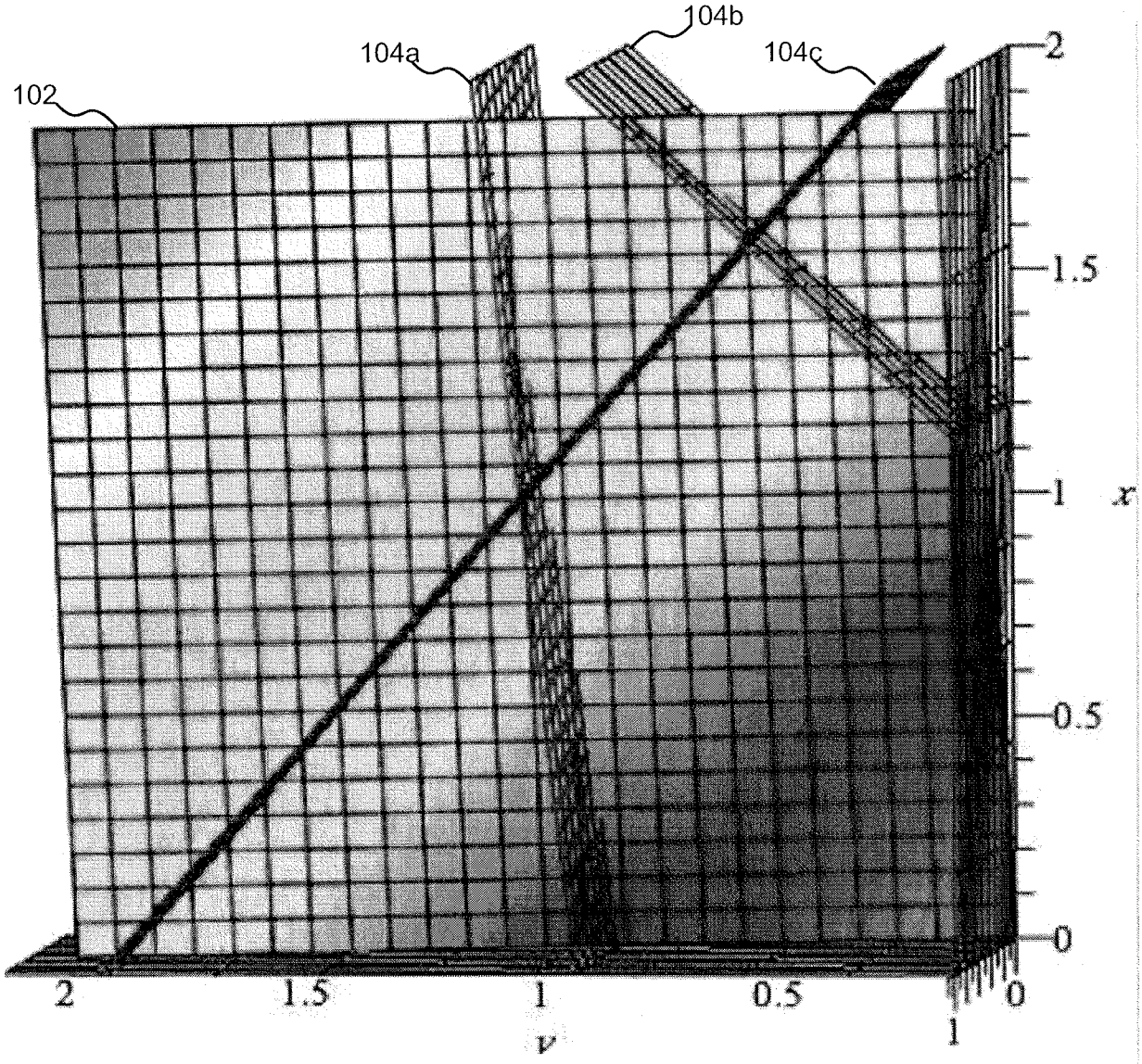

FIG. 1

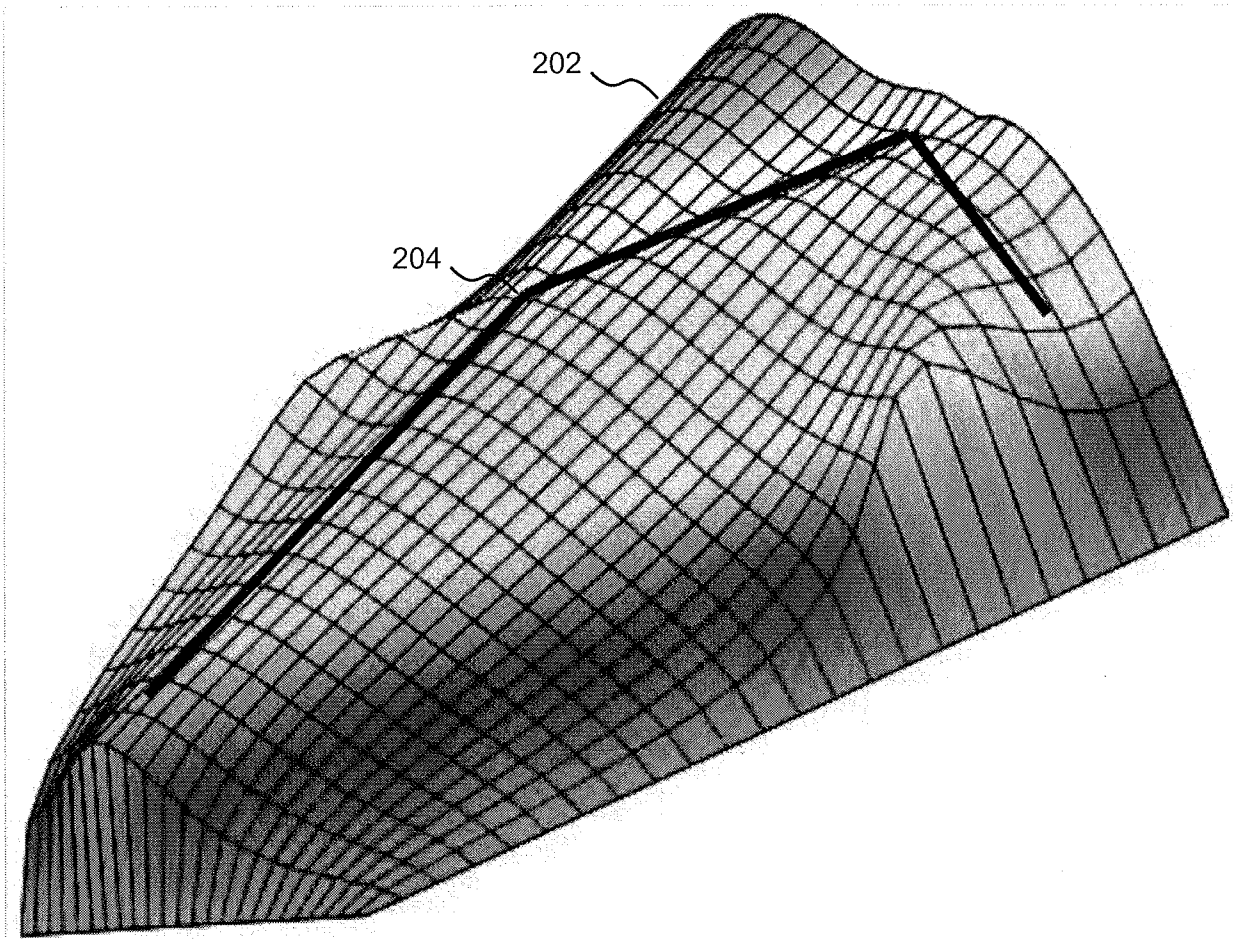

FIG. 2

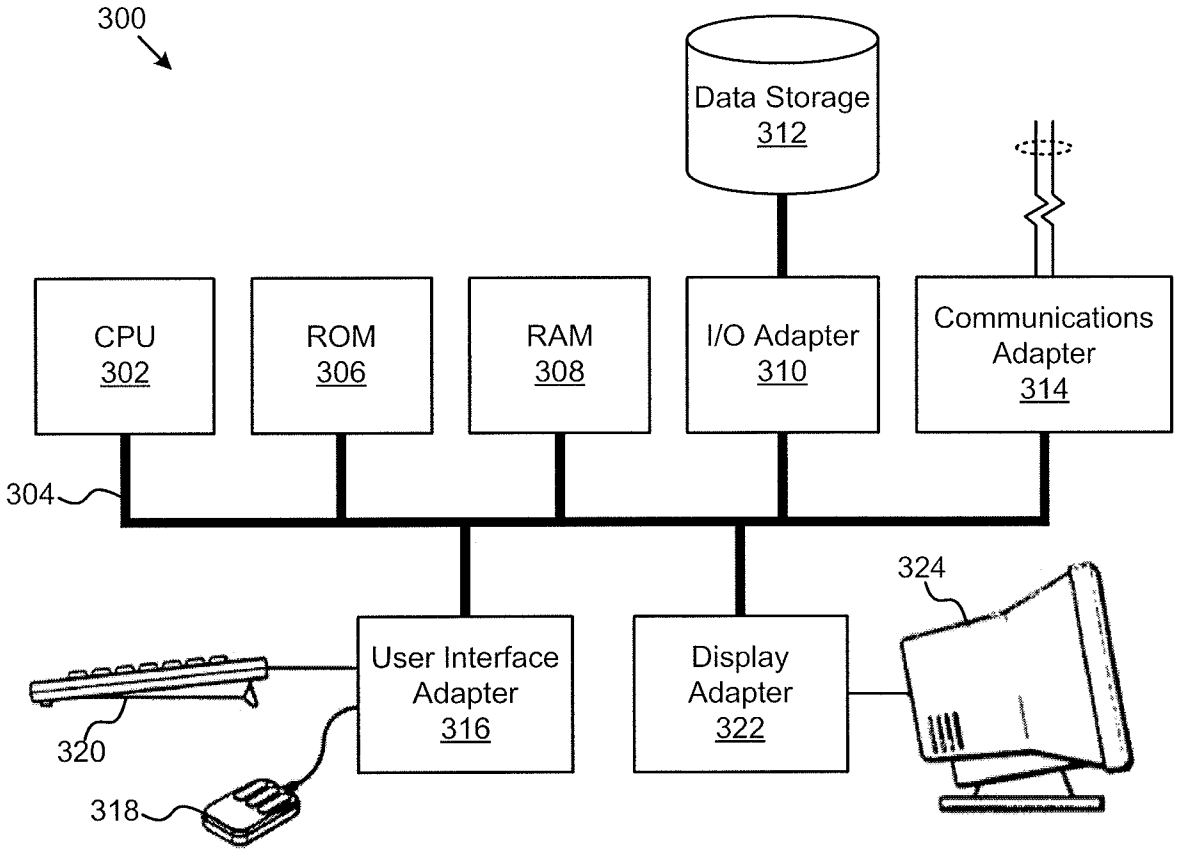
300
Data Storage
312
CPU
302
ROM
306
RAM
308
I/O Adapter
310
Communications
Adapter
314
304
User Interface
Adapter
316
Display
Adapter
322
324
320
318

FIG. 3

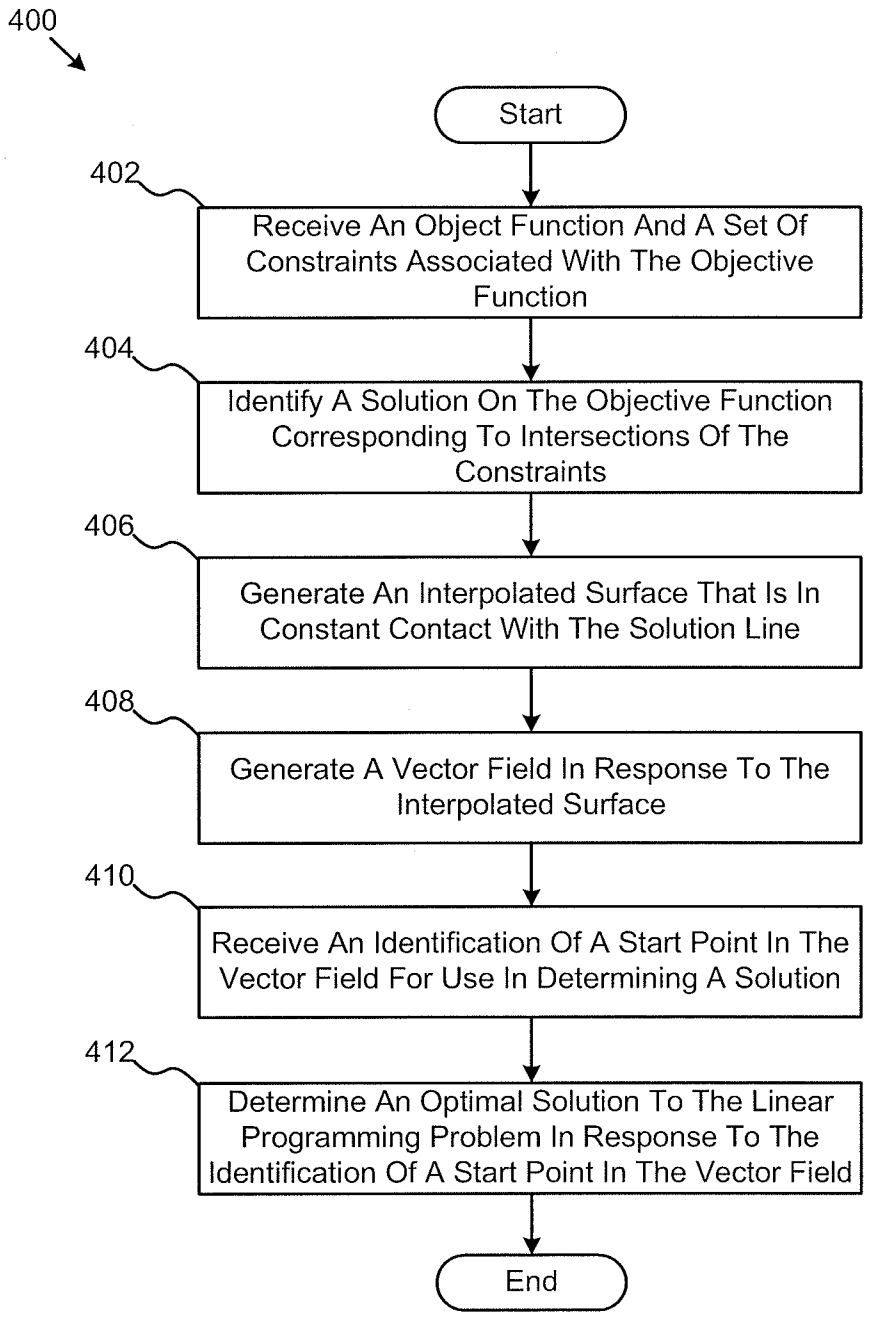
400
Start
402
Receive An Object Function And A Set Of Constraints Associated With The Objective Function
404
Identify A Solution On The Objective Function Corresponding To Intersections Of The Constraints
406
Generate An Interpolated Surface That Is In Constant Contact With The Solution Line
408
Generate A Vector Field In Response To The Interpolated Surface
410
Receive An Identification Of A Start Point In The Vector Field For Use In Determining A Solution
412
Determine An Optimal Solution To The Linear Programming Problem In Response To The Identification Of A Start Point In The Vector Field
End

FIG. 4

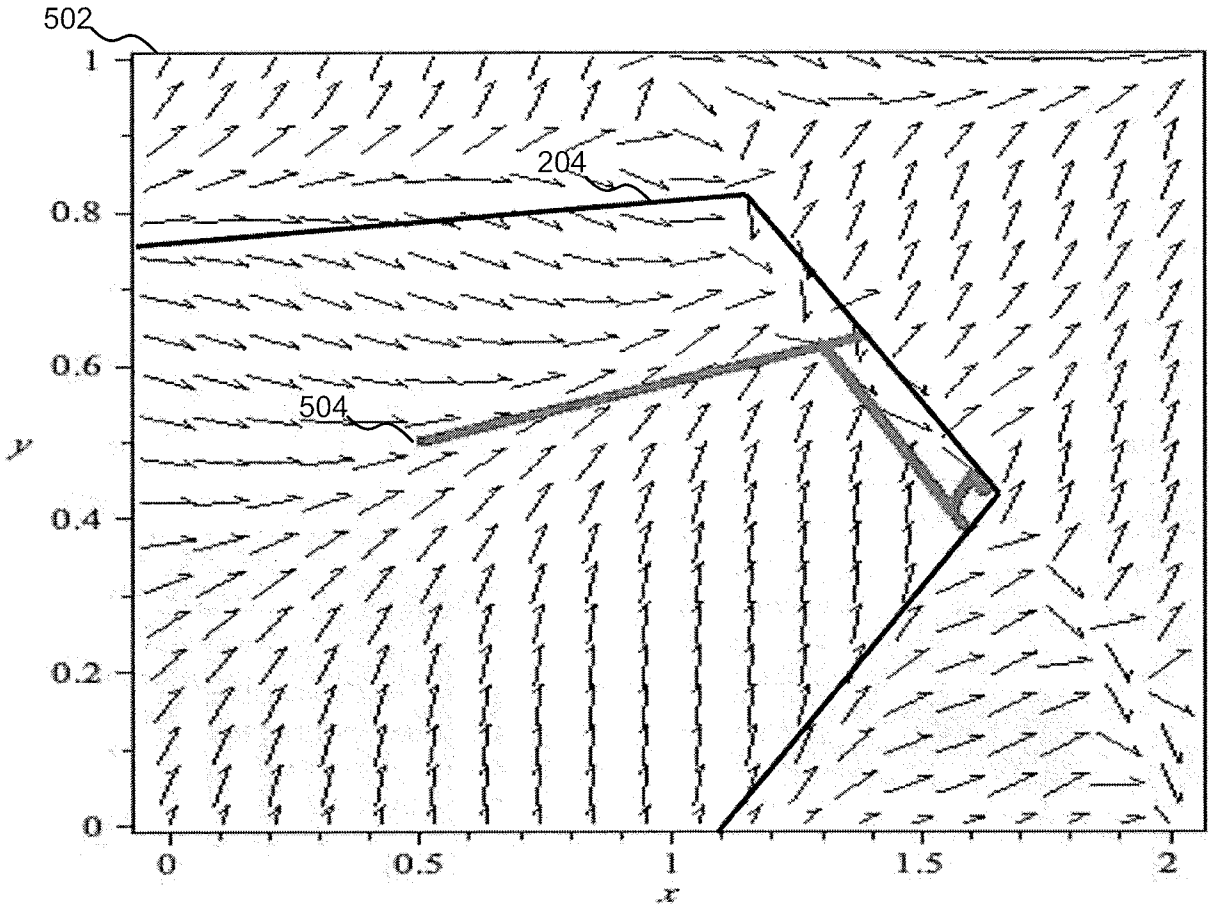

FIG. 5

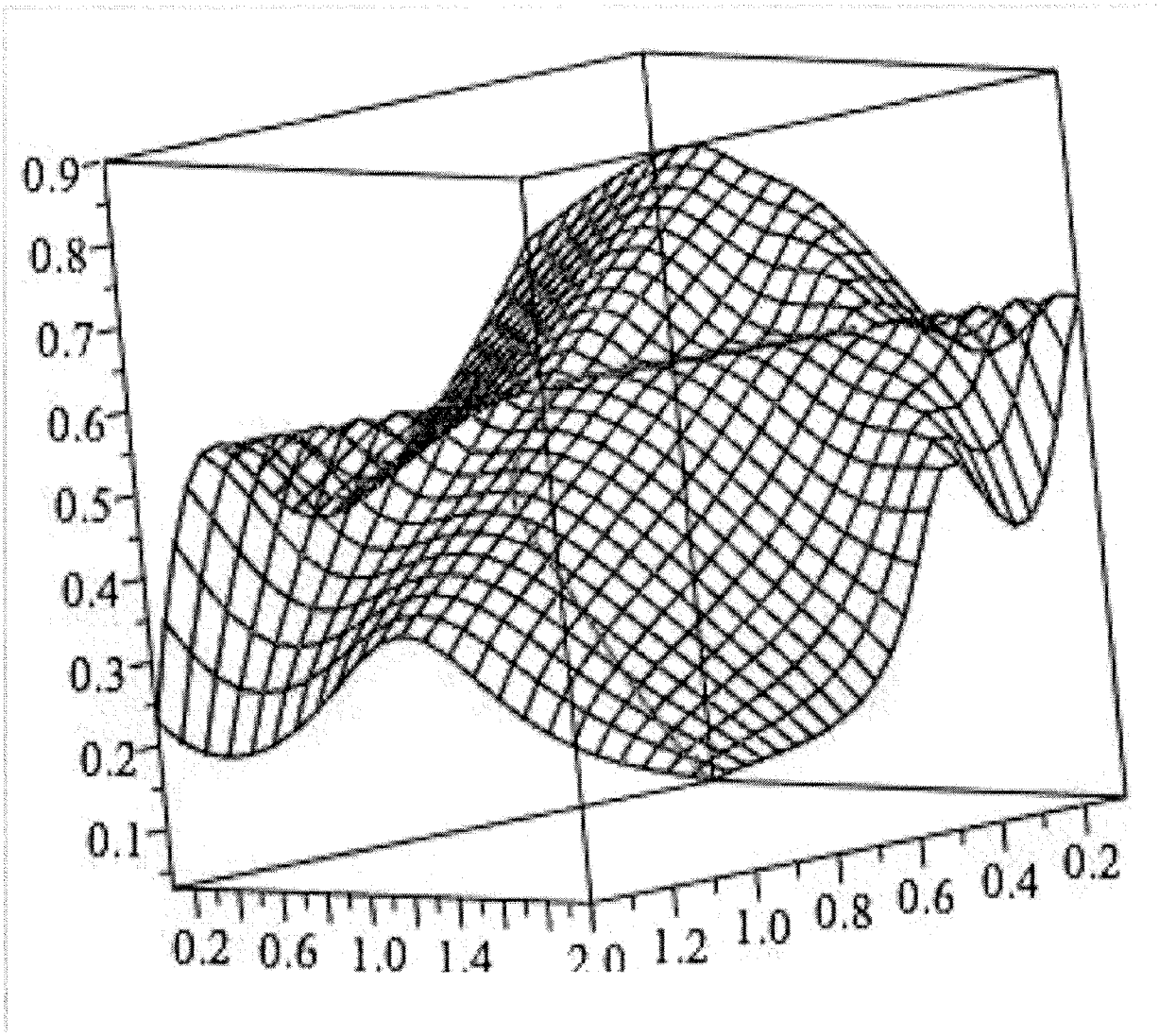

FIG. 6

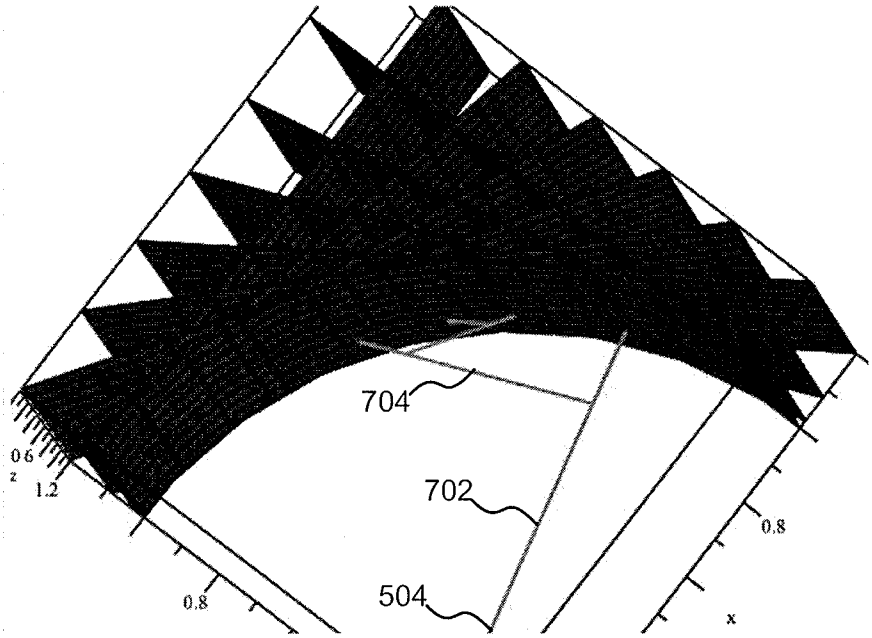

FIG. 7A

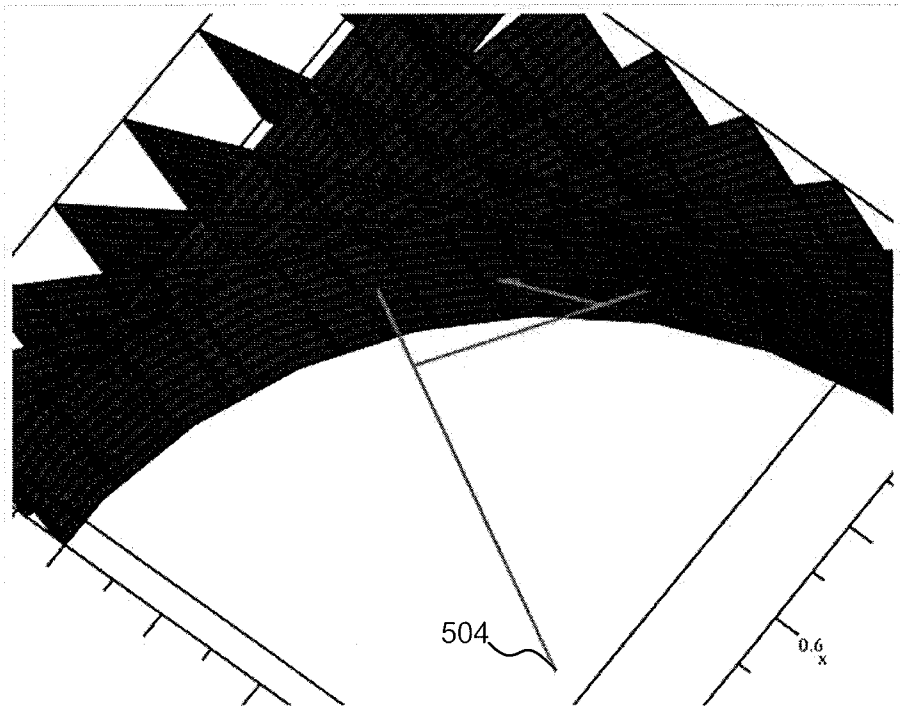

FIG. 7B

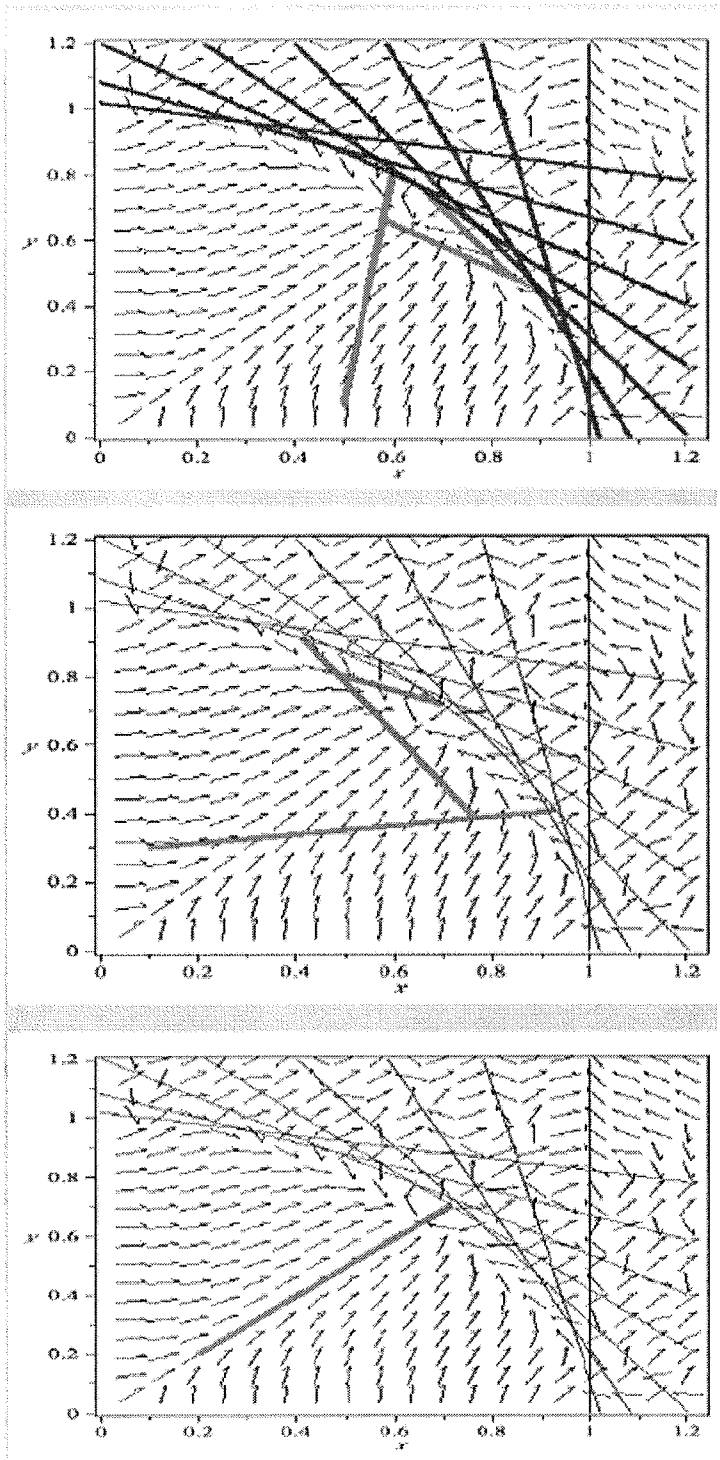

FIG. 8

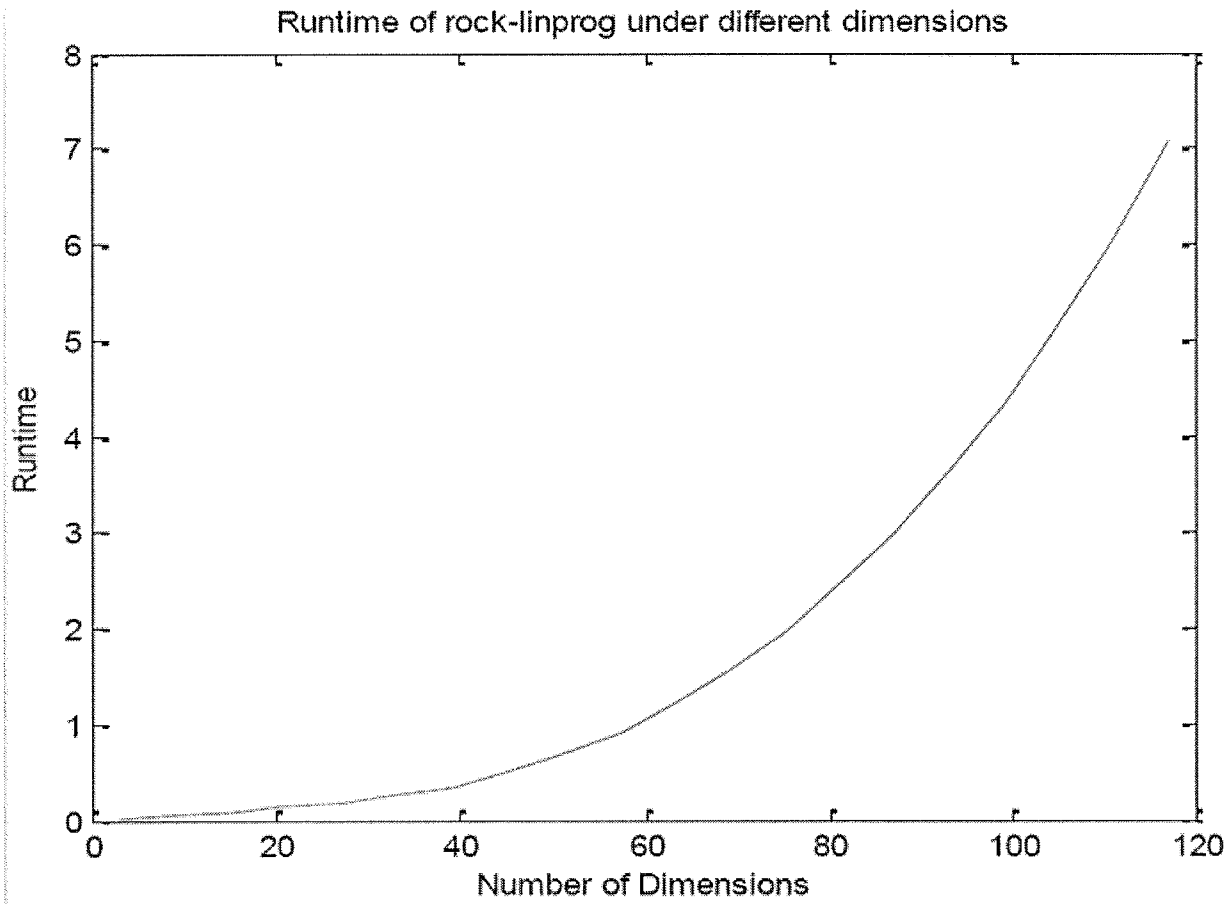
Runtime of rock-linprog under different dimensions
Runtime
Number of Dimensions
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100
120

FIG. 9

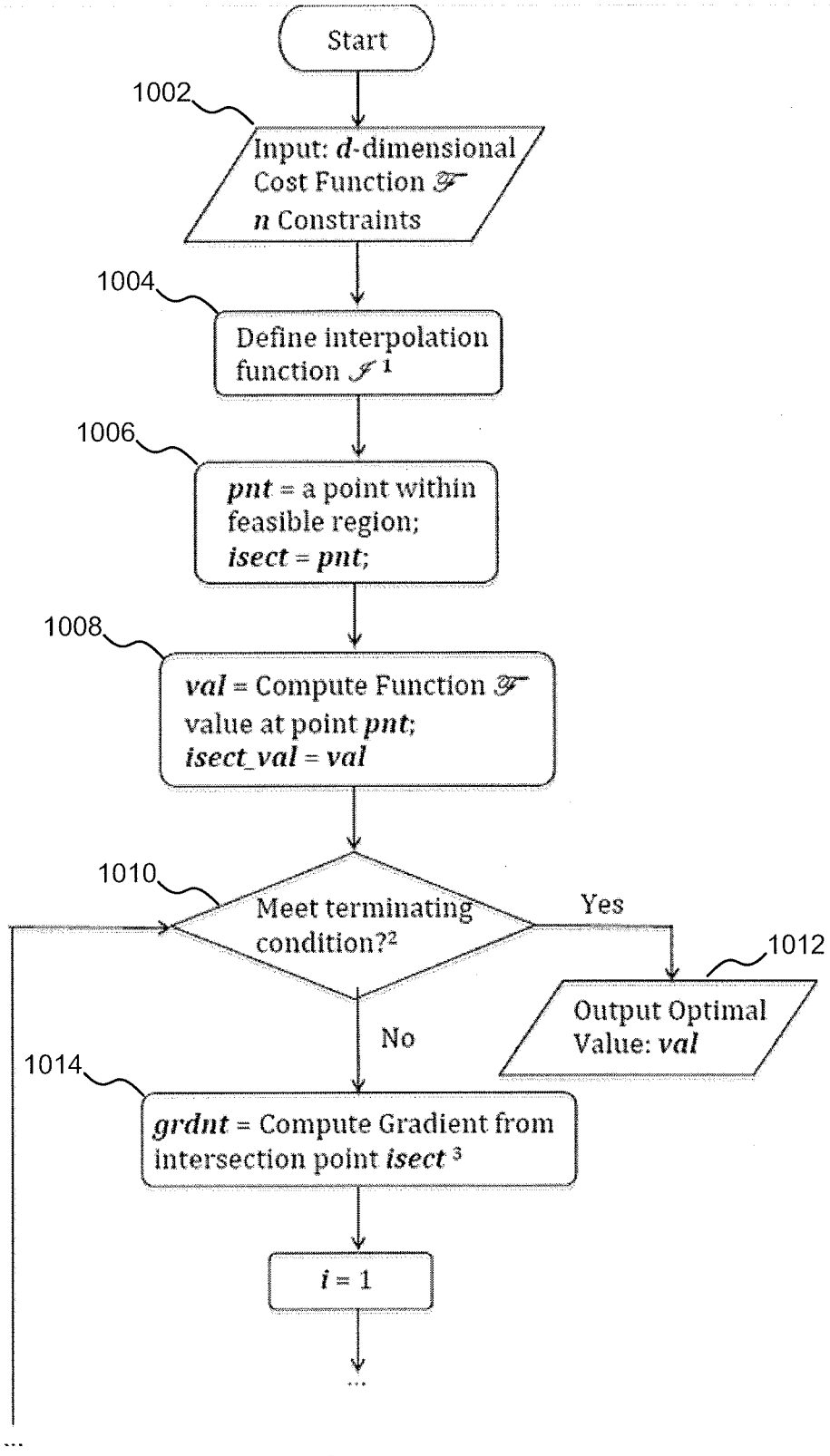
Start
1002
Input: d-dimensional
Cost Function
n Constraints
1004
Define interpolation
function 1
1006
pnt = a point within
feasible region;
isect = pnt;
1008
val = Compute Function
value at point pnt;
isect_val = val
1010
Meet terminating
condition?2
Yes
1012
Output Optimal
Value: val
No
1014
grdnt = Compute Gradient from
intersection point isect 3
i = 1

FIG. 10A

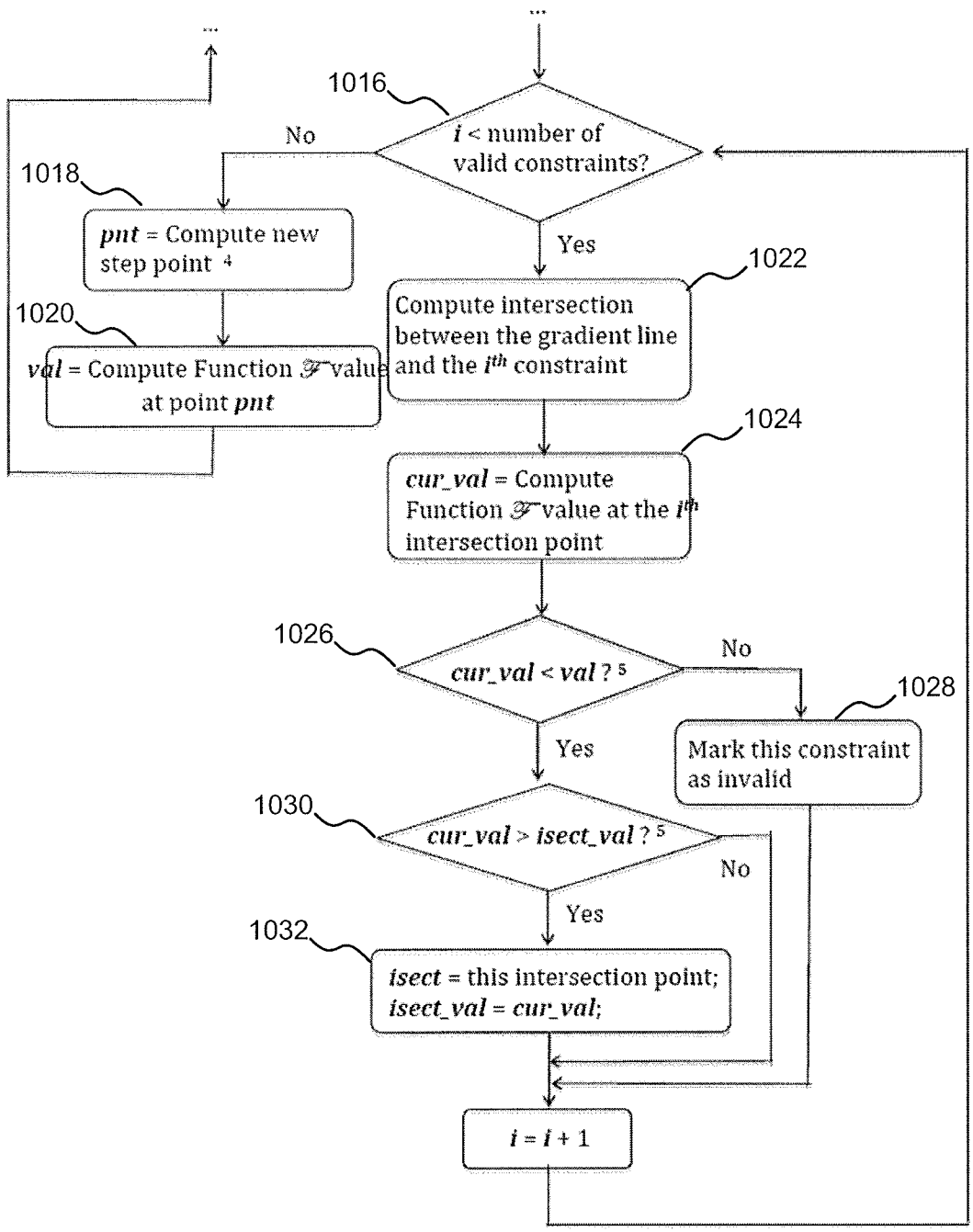
...
1016
i < number of valid constraints?
No
Yes
1018
pnt = Compute new step point 4
1020
val = Compute Function 𝓕 value at point pnt
1022
Compute intersection between the gradient line and the i^th constraint
1024
cur_val = Compute Function 𝓕 value at the i^th intersection point
1026
cur_val < val ? 5
No
Yes
1028
Mark this constraint as invalid
1030
cur_val > isect_val ? 5
No
Yes
1032
isect = this intersection point;
isect_val = cur_val;
i = i + 1

FIG. 10B

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/044334

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/11
ADD.

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)
G06F

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)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CHRISTOPHER A. HANE ET AL: "The fleet assignment problem: Solving a large-scale integer program", MATHEMATICAL PROGRAMMING, vol. 70, no. 1-3, 1 October 1995 (1995-10-01), pages 211-232, XP55042683, ISSN: 0025-5610, DOI: 10.1007/BF01585938 the whole document -----	1-18

☐ 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
14 November 2012	21/11/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Huguet Serra, G